TREBALL DE FI DE CARRERA

TÍTOL DEL TFC: Plataforma robòtica per navegació indoor

TITULACIÓ: Enginyeria Tècnica de Telecomunicació, especialitat Sistemes de Telecomunicació

AUTORS: Joana Castillo Perelló
Helena Medio Bonavida

DIRECTOR: Juan López Rubio

COODIRECTOR: Alex Albalà Díaz

DATA: 24 de febrer de 2014

Titol: Plataforma robòtica per navegació indoor

Autors: Joana Castillo Perelló
Helena Medio Bonavida

Director: Juan López Rubio

Coordirector: Àlex Albalà Díaz

Data: 24 de febrer de 2014

Resum

Aquest projecte és una ampliació i millora d'un projecte anomenat "Indoor and Outdoor Rover with simultaneous localization and mapping (SLAM)" el qual consisteix en un robot que al deixar-lo en un entorn qualsevol és capaç d'identificar-lo, amb l'ajuda de sensors i el mostra els resultats per pantalla a la estació base.

L'objectiu actual és millorar aquest projecte, per una part el moviment autònom del vehicle i, per altre, afegint nous mètodes per al control remot per part de l'usuari. La finalitat principal és la seva utilització per a la docència, poder treballar amb el robot a classe ampliant les seves funcionalitats i ajudar als alumnes a introduir-se al món de la programació robòtica. Abans de començar amb la introducció dels nous elements i mecanismes, es fa un petit resum del projecte anterior per entendre el seu funcionament. Un cop introduït aquest es van estudiant un a un els elements nous.

Per al moviment autònom s'han introduït tres punts nous: el magnetòmetre que permet una posició del robot més ajustada, el dispositiu Kinect que amplia el nombre de punts utilitzats com a referència central i, el mecanisme PID que permet un moviment més fluid. Paral·lelament, per al control remot es crea una aplicació mòbil i, també, s'afegeix el control a partir del dispositiu Leap Motion, el que permet moure el robot amb el moviment de la mà.

Per a cada nova aportació es fa un estudi individual i, un cop comprovat el seu bon funcionament individual, s'afegeix al conjunt del projecte amb el qual es realitzen una sèrie de proves i s'ajusta a les necessitats si és convenient.

Finalment, s'han realitzat una sèrie de tests per observar els resultats obtinguts en cada cas. D'aquesta manera es comprova que el funcionament és el esperat i, per tant, els objectius proposats es compleixen satisfactòriament.

Title: Robotic platform for indoor navigation.

Author: Joana Castillo Perelló
Helena Medio Bonavida

Director: Juan López Rubio

Coo-director: Àlex Albalà Díaz

Date: Fabruary, 24th 2014

Overview

This project is an extension and improvement of a project called “Indoor and Outdoor Rover with simultaneous localization and mapping (SLAM)” which consists on a robot that on having left in any environment is able to identifying it with the sensors and shows the results in the base station.

The current objective is to improve the previous project. One hand the independent movement of the robot and, the other hand, adding new methods for remote control for the user. The purpose is use in teaching, the student work with the robot in class and they can improve or implement more functionalities. Before starting with the introduction of new elements and mechanisms, there is a short summary of the previous project to understand its operation. After introducing this project is done a study of the new items.

For the independent movement are been introduced three new elements: the magnetometer which allows more accurate position of the robot, Kinect device that extends the number of point used as a central reference and the PID mechanism that allows for seamless movement. For the remote control, we are creating a mobile application, and also add control from Leap Motion device, which allows to the user move the robot with the movement of the hand.

For each new contribution, we are going to do an individual study and once verified its individual functioning is added to the overall project, with which they performed a series of tests and adjusted to the needs if necessary.

Finally, we are done different tests to observe the results in each case. In this way is checked that the result is as expected and therefore the proposed objectives are met satisfactorily.

ÍNDEX

LLISTA DE FIGURES

LLISTA DE TAULES

INTRODUCCIÓ

“Indoor and Outdoor Rover with Simultaneous Localization and Mapping (SLAM)” és un projecte que va realitzar Alex Albalà com a tesis final de màster. L’objectiu d’aquest era desenvolupar el hardware i el software d’un sistema SLAM de baix cost compost per dos subsistemes: un robot autònom de baix cost i una estació de terra on es visualitzen dades com telemetria i informació del robot. La finalitat d’un algoritme SLAM és poder deixar el robot en un entorn desconegut i que ell mateix identifiqui l’entorn mitjançant sensors i extregui un mapa aproximat.

Un cop finalitzada la tesis es van proposar millores respecte a elements existents com:

- Resolució de codificadors. La baixa resolució real dels codificadors introdueix petits errors però al ser acumulatius es converteixen en grans errors amb el temps.
- Ús de magnetòmetre o IMU. Mecanisme per conèixer la posició en tot moment.
- Més punts de detecció. Els sensors infrarojos instal·lats en el sistema tenen un baix nombre de punts de detecció. Amb un número més gran, l’algoritme SLAM es pot millorar i, inclús, es podria considerar correcta la posició del robot amb el tancament del bucle del sistema.
- Instal·lació de placa més potent. Amb això es podria implementar un algoritme SLAM en l’estació mòbil (rover) en lloc de en l’estació de terra, el que faria l’estació de terra més independent i, per tant, es tindria un sistema més modular.

A part d’aquestes millores, es va trobar una manca d’eines que permetessin a l’usuari interactuar de forma directa amb el robot. Una vegada arribat aquí, es decideix agafar aquest projecte i aplicar algunes de les millores proposades i afegir –ne de noves.

El primer que s’introduirà és un mecanisme de localització, un magnetòmetre. Amb aquest nou element es podrà obtenir la posició del robot amb exactitud, amb el que es vol aconseguir eliminar els errors introduïts pels encoders.

Un altre punt a tenir en compte és aconseguir un sistema amb un moviment més fluid. Per a això, es desenvoluparà un controlador PID per controlar el gir del robot i el seguiment de paret quan aquest vagi recte.

Per resoldre la manca de punts de detecció amb els sensors infrarojos s'introduirà un Kinect. Amb aquest dispositiu es vol substituir el sensor central i, d'aquesta manera, obtenir molts més punts de detecció.

Per últim, es desenvoluparan dos mètodes per al control remot del robot. Com a primer mètode, es controlarà el robot amb un dispositiu anomenat Leap Motion el qual permetrà el moviment a partir de la posició de la mà. I com a segon mètode es crearà una aplicació mòbil Android.

D'aquesta manera el treball quedarà organitzat de la següent manera:

- Capítol 1: Resum de la tesis principal
- Capítol 2: Magnetòmetre
- Capítol 3: Dispositiu Kinect
- Capítol 4: Controlador PID
- Capítol 5: Connexions utilitzades
- Capítol 6: Mètodes introduïts de control remot
- Capítol 7: Avaluació de resultats obtinguts
- Capítol 8: Conclusions i futures implementacions

CAPÍTOL 1. TREBALL PREVI

1.1 Introducció

En aquest primer capítol s'explicarà la part realitzada prèviament en forma introductòria per tal d'entendre què hi havia i quins seran els canvis que s'aplicaran més endavant.

Aquest projecte està basat en el funcionament d'un aspirador robòtic. L'aspirador inicia el moviment fent un espiral cada vegada més gran fins trobar un obstacle. Quan el troba, considera que és el perímetre del recinte i continua tot recte fins trobar un altre obstacle, moment en el qual, corregirà la seva trajectòria.

A diferència de l'aspirador, el rover detectarà i coneixerà la posició dels obstacles i, alhora, la seva pròpia posició. Amb l'ajuda del hardware que s'afegirà es podrà identificar la posició i la distància recorreguda.

Tal i com s'ha comentat, el rover tindrà dues tasques: la seva pròpia localització i el mapeig del recinte. Les dues tasques estan relacionades ja que no té sentit implementar un algoritme de mapeig sense la part de localització i viceversa. Combinant les dues tasques existeix el que s'anomena algoritme SLAM (Simultaneous Localitzation And Mapping).

El mapa es construeix a partir d'una escala de grisos per localitzar la distància a la que es troben els obstacles. El blanc vol dir que no hi ha obstacle i el negre vol dir que la probabilitat de que hi hagi un obstacle és del 100%.

Aquest algoritme està dissenyat com una manera de combinar la informació obtinguda a partir dels valors de sensors làser i la informació extreta de la odometria. Per tant, s'haurà de modificar aquest algoritme ja que s'utilitzaran sensor d'infrarojos.

L'odometria és l'estimació de la posició a partir de les dades obtingudes dels diferents sensors. Aquesta posició no serà correcta pels errors durant la propagació, l'error del propi càlcul i pel canvi de posició del rover just després de fer el càlcul.

Les dades s'obtindran dels encoders. Els encoders permeten conèixer la posició de les rodes, no en valor absolut, però sí quant han girat i la distancia que han avançat.

Hi ha dues maneres de realitzar el gir:

- Frenant una banda del rover i moure l'altre costat: el rover girarà cap al costat bloquejat (**Fig. 1.1**). L'avantatge d'aquesta forma de gir és que no

depens del centre de gravetat però té dos inconvenients: necessita molt espai per girar i al aturar la roda es crea un punt de contacte amb el terra desconegut i això afegeix un error.

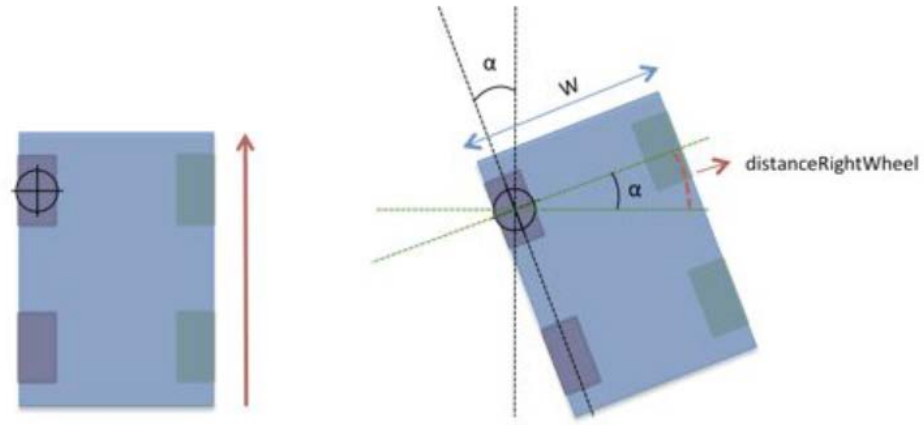


Fig. 1.1 Opció 1 de gir

- Moure una banda en una direcció i l'altre en direcció contrària: el rover girarà sobre el seu centre de gravetat (**Fig. 1.2**). El desavantatge més important d'aquest segon cas és que moltes vegades el centre de gravetat no coincideix amb el centre geomètric del rover però, com a avantatge té que és més ràpid girant i necessita menys espai.

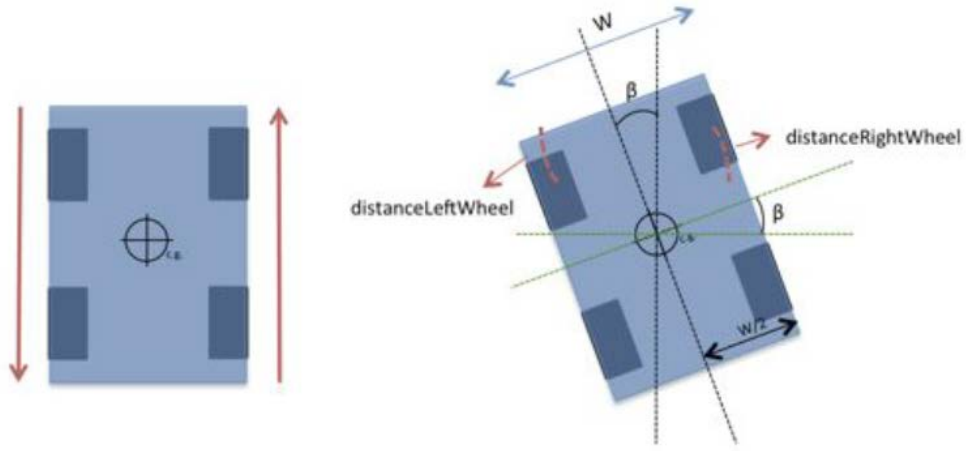


Fig. 1.2 Opció 2 de gir

Tots dos mètodes introdueixen errors per això cal trobar la manera de reduir-los. Per tal de mesurar els errors, es desenvoluparà el [1] *Bidirectional Square Path Experiment* (UMB-mark).

1.2. Implementació

En aquest apartat s'explicarà l'estructura software i quins han sigut els components escollits amb una breu descripció de les seves característiques.

1.2.1. Implementació hardware

Abans de llistar els diferents components utilitzats, s'exposarà, en una visió general, quina serà l'estructura (**Fig. 1.3**).

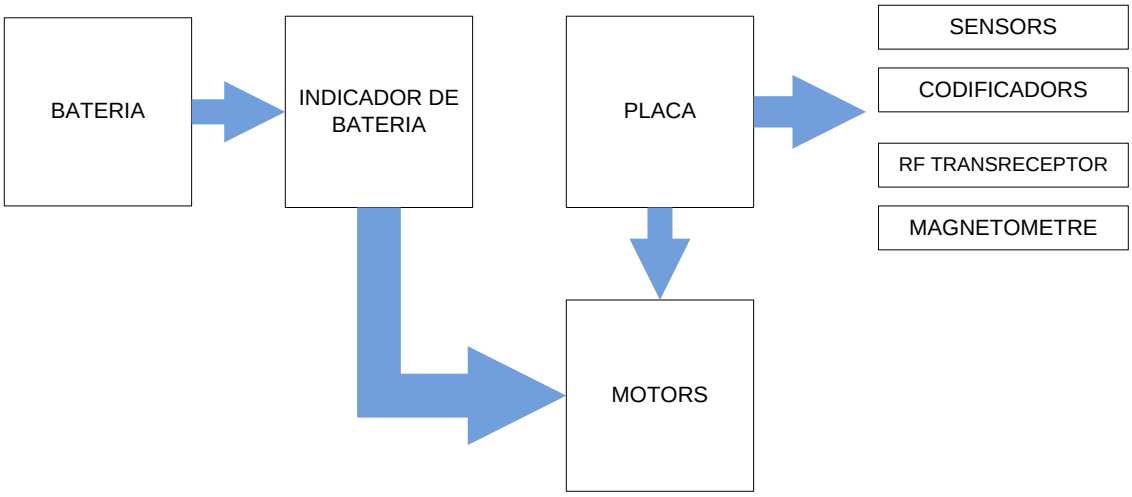


Fig. 1.3 Diagrama de blocs

La placa serà l'encarregada de processar tota la informació obtinguda dels diferents components i, a través del mòdul de comunicació, estarà en comunicació continua amb l'estació base. La bateria i el indicador de bateria formen el bloc de subministrament d'energia. Per assegurar el bon funcionament de la bateria, s'introduirà un indicador que saltarà quan aquesta estigui a punt d'esgotar-se i els motors deixaran de funcionar immediatament. L'equip seleccionat és DFRobot 4WD Arduino-Compatible Platform w/ Encoders de RobotShop. Tal i com es mostra a la figura (**Fig. 1.4**), el kit inclou:

- L'estructura externa (230x185x110mm).
- 4 rodes: 65 mm de diàmetre
- 4 Motors DC
- 2 codificadors
- Interruptor d'alimentació

Fig. 1.4 Kit DFRobot 4WD

Per al control dels motors és necessari un controlador de motor continu. El controlador ZX-DCM2 DC Motor (**Fig. 1.5.**) té dos canals amb un indicador LED de dos colors, tensió d'alimentació de +4.8 a 30 Vdc i connectors PCB INEX Standard 3-pin.

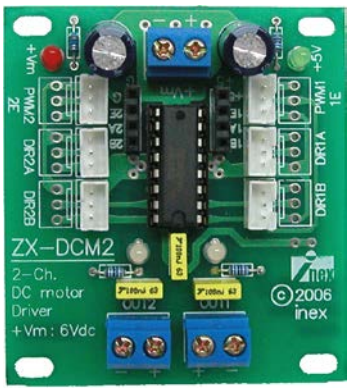


Fig. 1.5 ZX-DCM2

Com a placa principal s'ha escollit la Fez Panda pel baix cost i per les característiques que ofereix. Fez Panda (**Fig. 1.6**) és una placa de dimensions petites que utilitza Microsoft .NET Micro Framework, accepta qualsevol tipus de shield Arduino i inclou gran varietat de llibreries.

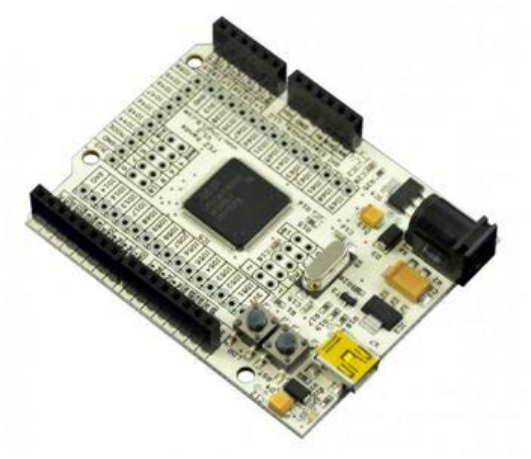


Fig. 1.6 Fez Panda

La part de mapatge del Rover necessita sensors que recullin la informació del entorn i l'enviïn a la placa. Els sensors escollits són 4 sensor SHARP infrarojos petits, de baix consum i de baix cost (**Fig.1.7**).

Fig. 1.7 Sensor SHARP

Pels sensors davanters s'ha escollit el model SHARP GP2Y0A21 amb un abast efectiu de 10 a 80cm. Pels sensors laterals, el model és SHARP GP2D120 amb un abast efectiu de 4 a 30cm. Ambdós tenen una sortida de tensió analògica.

Els encoders són uns elements per la auto-localització del rover. És un sensor digital localitzat als motors que permet conèixer la posició actual del motor. Com a mòdul de comunicació s'ha escollit els mòduls XBEE. Aquests mòduls són components hardware que implementen l'estàndard IEEE 802.15.4.8. Treballen amb una corrent d'alimentació baixa i ofereixen un ample de banda de transmissió de 250kbps de treball a la banda de 2,4GHz.

Després de realitzar un estudi del consum de tots els components, la bateria seleccionada ha sigut una Fullymax amb les següents característiques:

- Voltatge: 11,1 V
- Intensitat: 4000mAh
- 3SP1 (3 cel·les en sèrie i 1 en paral·lel)
- Bateria de polímer de ions de liti.

Per últim, es mostra un resum de les connexions hardware de tot el sistema complet (**Fig. 1.8**). Les línies negre i vermella són la tensió d'alimentació i el blau, el control.

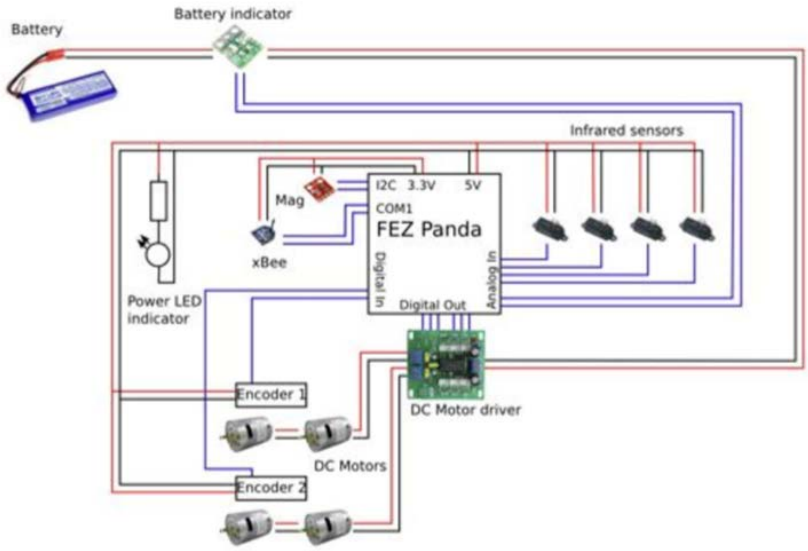


Fig. 1.8 Esquema general del sistema

1.2.1. Implementació software

La solució s'ha implementat en llenguatge C# amb framework .NET Micro Framework versió 4.1. A continuació, es representa, sense entrar molt al detall, l'estructura completa del software (**Fig. 1.9**).

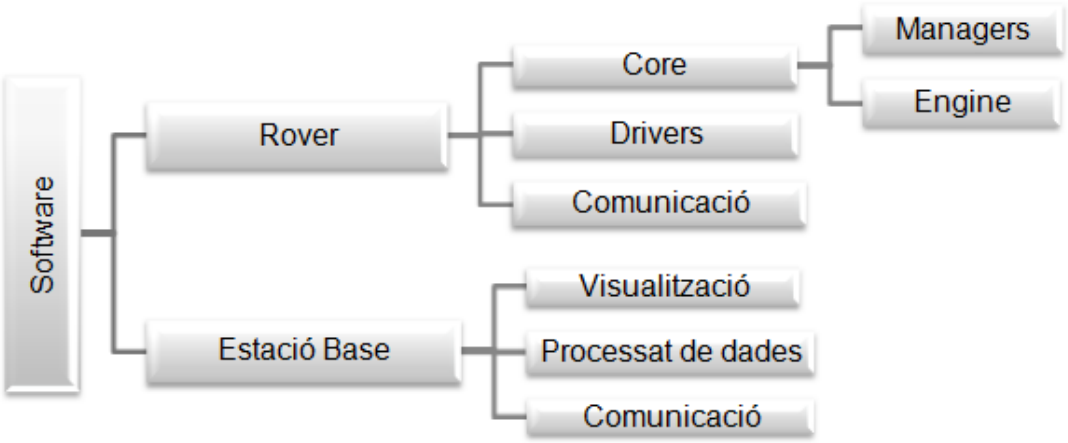


Fig. 1.9 Diagrama de blocs

El codi està dividit en dos grans blocs: rover i estació base. Cadascun d'aquest blocs està dividit en d'altres més petits que interactuen entre ells.

El *Core* està composat pels *Managers* que s'encarreguen de gestionar i l'*Engine* que conté tota la part lògica del moviment, té accés a la connexió ràdio i gestiona la informació rebuda dels *Managers*.

El mòdul del *Core* està dividit en 4 parts:

- **Navigation manager:** inclou tots aquells mètodes que tinguin relació amb el control de moviment del vehicle tant si és manual com automàtic.
- **Battery manager:** inclou tots els mètodes per conèixer l'estat de la bateria.
- **Contigency:** mòdul que controla qualsevol incidència que pugui succeir ja sigui de conducció com de l'estat dels components.
- **Engine:** Controla totes les parts del *Core* i, a més a més, gestiona la comunicació amb l'estació base.

Els drivers són els encarregats de proporcionar una interfície per interactuar amb els sensors. Es defineix un driver per cada element dels que s'extreuen dades:

- **Infrared sensor driver:** Serà anomenat *DistanceDetector* i realitza la conversió del voltatge de sortida a distancia en centímetres.
- **Encoders driver:** Comptabilitza els *ticks* del encoder i ho transforma en distancia recorreguda. Aquest procés el realitza per a cada roda de manera independent.
- **Motors driver:** La tasca d'aquest driver és comunicar-se amb el driver DC Motor per tenir un control dels motors.
- **Magnetometer driver:** Rep i envia les coordenades de posició i fa les conversions pertinents segons les necessitats.
- **Battery indicator driver:** s'anomena *VoltageCurrentSensor* i s'encarrega de convertir les sortides anàlogues en valors de voltatge i corrent segons especificacions del datasheet.

La part de la comunicació està dividida en dues parts: codificació i transport. La capa de codificació és l'encarregada de la serialització de les dades i la implementació del protocol propietari per a la comunicació interna. En canvi, la capa de transport s'encarrega de la connectivitat i de la transmissió de dades.

L'estació base és una aplicació visual basada en la tecnologia WPF (*Windows Presentation Foundation*). També està programada en C#. En aquesta pantalla es representarà la ruta del rover, diversos paràmetres per conèixer l'estat del sistema, el sensor que detecta obstacles, entre d'altres paràmetres (**Fig. 1.10**).

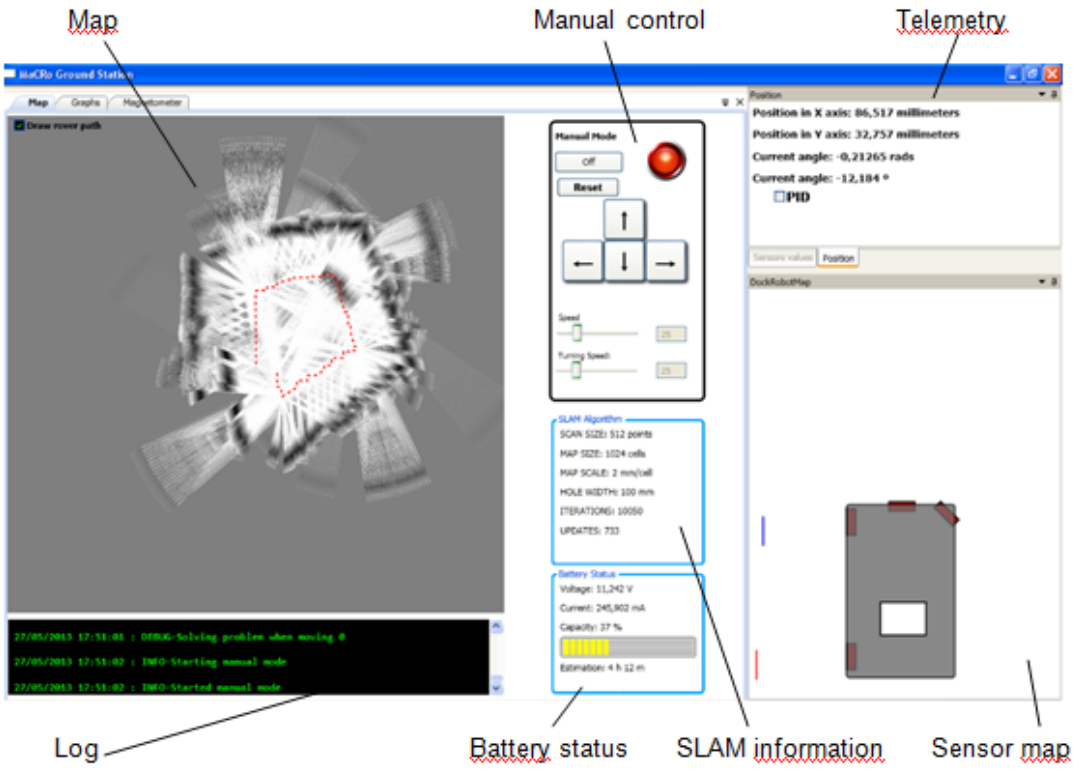


Fig. 1.10 Estació base

A partir de tot el que es té es decideix millorar i ampliar el robot, per a la seva posterior utilització a les classes. D'aquesta manera es podrà mostrar als alumnes i aquests podran utilitzar-lo per a estudiar-lo, programar-lo, crear aplicacions per a controlar-lo, etc. sense la necessitat de començar de zero.
Aquest tipus de robot o semblants ja existeixen al mercat, però el seu preu és molt més elevat que el que s'ha creat en el projecte.

1.3. Conclusions

Els components escollits s'han basat en un estudi previ on s'ha tingut en compte les necessitats del projecte i les limitacions econòmiques establertes.
El software desenvolupat és senzill i modular la qual cosa permet fàcilment introduir nous elements. El processat de les dades s'ha intentat fer de manera que no comportés una gran càrrega i així es realitzés de forma ràpida.

Tota la informació exposada en aquest capítol es pot veure de forma ampliada a la tesis *Indoor and Outdoor Rover with Simultaneous Localization and Mapping (SLAM)* (veure [1]).

CAPITOL 2. MAGNETOMETRE

2.1 Introducció

El magnetòmetre és un dispositiu que ens permet obtenir, amb exactitud, la posició del rover mitjançant als camps magnètics de la Terra. Ens permet conèixer, tanmateix, l'angle de gir comparant el valor actual donat pel magnetòmetre amb la posició inicial abans d'arrancar els motors. Amb la introducció d'aquest nou hardware s'eliminen els errors acumulatius introduïts pels encoders.

Fins ara hi havia una gran dependència amb el sòl on es feien les proves ja que els encoders afegeixen un error que varia segons el material del terra. Un valor afegit que ens proporciona la implementació del magnetòmetre és que ja no dependrà d'on es facin les proves perquè els errors acumulatius introduïts pels encoders quedaran minvats pel magnetòmetre.

El magnetòmetre utilitzat serà el MAG3110 (veure [2]). Dóna informació dels 3 eixos (x,y,z) de forma digital però, en aquest cas, només s'han implementat els eixos x i y. Està format per quatre pins: dos per l'alimentació (1.95 - 3,6V) i dos per l'obtenció de les dades (SLC i SDA).

El bus de comunicació utilitzat és el I2C (**Fig. 2.1**). És un bus de comunicació sèrie el qual utilitza tres línies per enviar la informació: una de les línies és per les dades anomenat SDA, l'altre és pel rellotge per tal que estigui sincronitzat, SCL, i, per últim, la terra/massa.

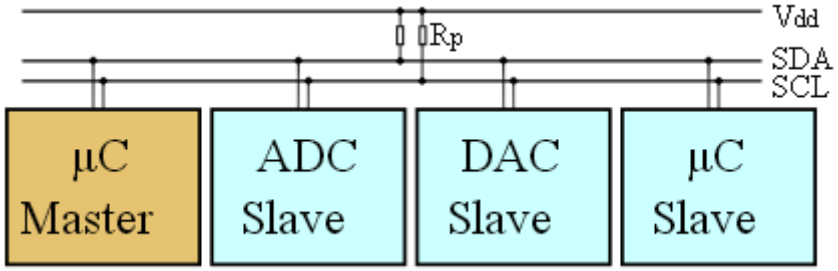


Fig. 2.1 Estació base

Cadascun dels dispositius connectats al bus tenen una adreça. Els dispositius poden ser de dos tipus: mestres o esclaus. Qualsevol dispositiu pot ser mestre però només pot haver-hi un. El mestre comença la comunicació enviant un patró anomenat *start condition* avisant als esclaus d'una nova transacció. A continuació, el mestre envia, amb un byte (A7-A1), l'adreça de l'esclau amb el qual vol mantenir comunicació. Tots els esclaus comparen aquesta adreça amb la seva pròpia i, si coincideix, l'esclau envia un ACK acceptant la comunicació. Amb el bit A0 el mestre indica l'operació desitjada, és a dir, si el bit és igual a 0 el mestre envia i si és igual a 1 el mestre rep. Seguidament, s'inicia la

comunicació entre els dos dispositius. Tots els bytes de dades han de tenir 8 bits. Cada byte llegit/escrit ha de ser acceptat per un ACK. El SCL és controlat pel mestre, però, ocasionalment, l'esclau pot canviar aquest paràmetre. Quan finalitzi la comunicació, el mestre envia una *stop condition* per alliberar el bus.

Taula 2.1 Trama de comunicació sèrie I2C

Start	A7A6A5A4A3A2A1	A0(R/W)	ACK	...DATA...	ACK	Stop	Idle

2.2. Calibratge

El magnetòmetre és molt sensible als camps magnètics del seu voltant i pot donar-se el cas que els valors obtinguts no siguin els correctes. Per tal de poder assegurar que aquests valors siguin fiables, es calibrarà abans de fer qualsevol prova. Es crea un programa diferent per fer el codi de calibratge que, més tard, s'inclourà al projecte.

El calibratge consisteix en realitzar un gir de 360 graus i mostrar el resultat per pantalla (**Fig. 2.2**).

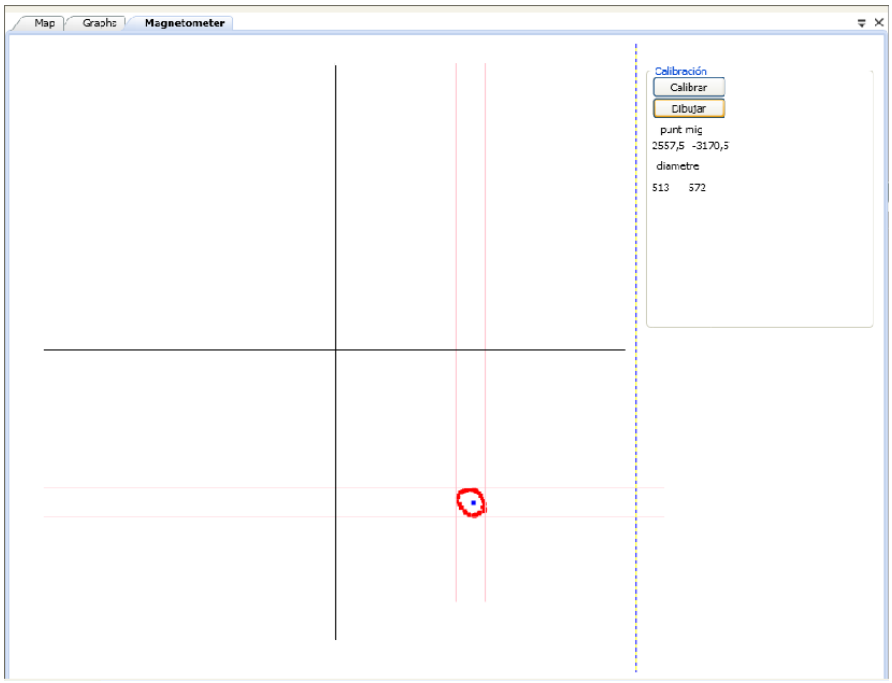


Fig. 2.2 Valors sense calibratge

El rang del magnetòmetre va de -3000 a 3000. En aquest cas, el rang dels valors obtinguts és de -1000 a 1000 aproximadament. Per tant, per aconseguir que el dibuix sigui visualment fàcil de veure, s'ha adaptat la pantalla realitzant

un simple càlcul. També s'ha fet una modificació dels valors del l'eix de les y ja que aquest són negatius. Totes aquestes adaptacions només s'implementen per la seva visualització i així es manté l'orientació coneguda dels eixos.

Un cop representats tots els punts obtinguts es calcula l'offset. Aquest offset és el valor que indica quina és la variació que provoquen els camps magnètics. Aquest valor és el resultat de la mitja del valor mínim i del màxim de cada eix. Un cop fet el càlcul, l'offset s'introdueix dins del magnetòmetre i es torna a fer la mateixa prova. Després de realitzar un gir de 360º, el resultat és el que es pot observar a continuació (**Fig. 2.3**).

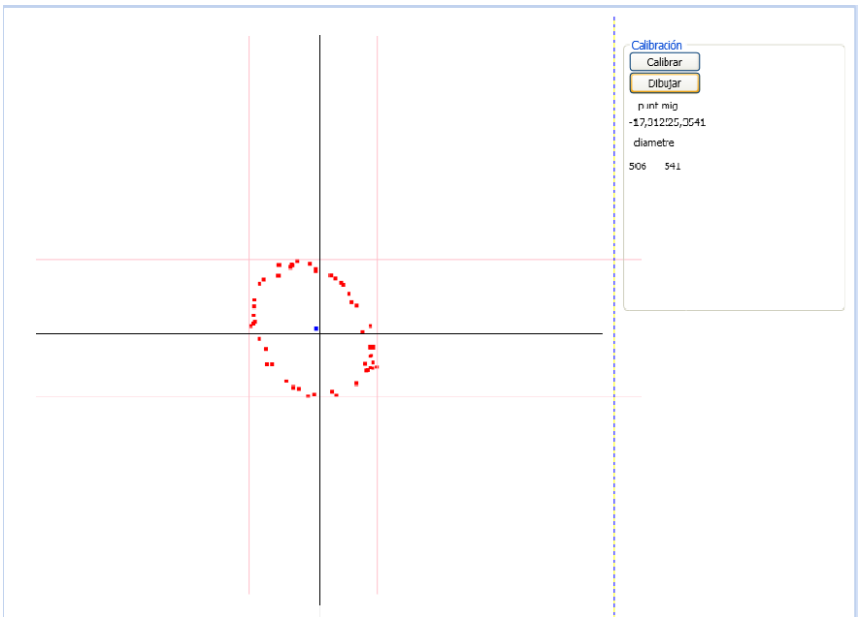


Fig. 2.3 Valors amb calibratge

2.3. Implementació

Un cop solucionat el tema del calibratge, s'uneix el magnetòmetre a la resta del projecte. Per tal de completar la implementació correctament s'hauran de realitzar alguns canvis en els registres. A continuació, s'explicaran tots els canvis realitzats. L'adreça utilitzada per la comunicació és 0x0E a una freqüència de 80Hz.

La **Taula 2.2.** mostra el mapa de les adreces dels registres.

Taula 2.2 Adreces de registre.

Name	Type	Register Address	Auto-Increment Address (Fast Read)(1)	Default Value	Comment
DR_STATUS(2)	R	0x00	0x01	0000 0000	Data ready status per axis
OUT_X_MSB(2)	R	0x01	0x02 (0x03)	data	Bits [15:8] of X measurement
OUT_X_LSB(2)	R	0x02	0x03	data	Bits [7:0] of X measurement
OUT_Y_MSB(2)	R	0x03	0x04 (0x05)	data	Bits [15:8] of Y measurement
OUT_Y_LSB(2)	R	0x04	0x05	data	Bits [7:0] of Y measurement
OUT_Z_MSB(2)	R	0x05	0x06 (0x07)	data	Bits [15:8] of Z measurement
OUT_Z_LSB(2)	R	0x06	0x07	data	Bits [7:0] of Z measurement
WHO_AM_I(2)	R	0x07	0x08	0xC4	Device ID Number
SYSMOD(2)	R	0x08	0x09	data	Current System Mode
OFF_X_MSB	R/W	0X09	0x0A	0000 0000	Bits [14:7] of user X offset
OFF_X_LSB	R/W	0X0A	0X0B	0000 0000	Bits [6:0] of user X offset
OFF_Y_MSB	R/W	0X0B	0X0C	0000 0000	Bits [14:7] of user Y offset
OFF_Y_LSB	R/W	0X0C	0X0D	0000 0000	Bits [6:0] of user Y offset
OFF_Z_MSB	R/W	0X0D	0X0E	0000 0000	Bits [14:7] of user Z offset
OFF_Z_LSB	R/W	0X0E	0X0F	0000 0000	Bits [6:0] of user Z offset
DIE_TEMP(2)	R	0X0F	0X10	data	Temperature, signed 8 bits in °C
CTRL_REG1(3)	R/W	0X10	0X11	0000 0000	Operation modes
CTRL_REG2(3)	R/W	0X11	0x12	0000 0000	Operation modes

És necessari conèixer periòdicament la posició, per això s'ha d'activar del Registre de Control (CTRL_REG1) en bit AC. Aquest bit s'identifica amb el bit 0 del registre (**Taula 2.3**).

Taula 2.3 Registre CRTL_REG1

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
DR2	DR1	DR0	OS1	OS0	FR	TM	AC

Per defecte el bit 0 està a zero, és a dir, en mode STANDBY.

Del Control de Registre 2 (CTRL_REG2) no és necessari activar res ja que per defecte tot està a zero. En aquest registre es controlen els reinicis automàtics i les correccions de les sortides (**Taula 2.4**).

Taula 2.4 Registre CRTL_REG2

Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0
AUTO-MRST-EN	-	RAW	Mag_RST	-	ST_Z	ST_Y	ST_X

Inicialment, els offset estan a zero ja que segons el lloc on es facin les proves els camps magnètics variaran. Cal calibrar abans de fer cap prova.

Com ja s'ha explicat a l'apartat anterior, un cop calculat l'offset el propi magnetòmetre fa l'adaptació. S'ha de tenir compte amb el bit 0 del registre

LSB. Aquest bit està a 0 perquè el registre sap que l'eix y és negatiu. Per tal de resoldre aquest petit problema, un cop s'han convertit els offset a bits, en el offset de l'eix Y, es traslladen els 16 bits una posició cap a l'esquerra per tal de prescindir del bit 15 que és el que dona el signe. A les taules següents es mostra el registre.

Taula 2.5 Registre OFF_X_MSB

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
XD14	XD13	XD12	XD11	XD10	XD9	XD8	XD7

Taula 2.3.6 Registre OFF_X_LSB

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
XD6	XD5	XD4	XD3	XD2	XD1	XD0	0

Taula 2.7 Registre OFF_Y_MSB

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
YD14	YD13	YD12	YD11	YD10	YD9	YD8	YD7

Taula 2.8 Registre OFF_Y_LSB

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
YD6	YD5	YD4	YD3	YD2	YD1	YD0	0

Taula 2.9 Registre OFF_Z_MSB

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
ZD14	ZD13	ZD12	ZD11	ZD10	ZD9	ZD8	ZD7

Taula 2.10 Registre OFF_Z_LSB

Bit 7	**Bit 6**	**Bit 5**	**Bit 4**	**Bit 3**	**Bit 2**	**Bit 1**	**Bit 0**
ZD6	ZD5	ZD4	ZD3	ZD2	ZD1	ZD0	0

Un del problemes més importants és que el magnetòmetre és molt sensible als camps magnètics puntuals. Tot i que es calibri, aquests camps magnètics provoquen errors de magnitud elevada. L'objectiu d'implementar el magnetòmetre era poder realitzar els girs a partir dels valors d'aquest sense verificar que siguin correctes, és a dir, considerar aquest valors com a absoluts.

La primera opció va ser guardar el valor inicial abans de començar el moviment i anar comparant constantment la posició actual amb la inicial. Fent això, els valors del magnetòmetre es converteixen en relatius (igual que els dels encoders).

La millor opció trobada ha sigut fer comparacions entre els valors dels encoders i els del magnetòmetre. Els encoders tenen un error mecànic de 18º. Si el valor del magnetòmetre és major que el sumatori entre el valor de l'encoder més 9º, es suposarà que el valor del magnetòmetre és incorrecte i, per tant, s'actualitzarà la posició actual amb la mitja de l'encoder i el magnetòmetre. En cas contrari, el valor del magnetòmetre i el de l'encoder són semblants, la posició actual s'actualitzarà amb el valor de l'encoder.

2.4. Conclusions

Amb la implementació del magnetòmetre s'han eliminat considerablement els errors acumulatius dels encoders. Al calibrar només es tenen en compte els camps magnètics estàtics. Si durant el recorregut hi ha una zona on el camp magnètic augmenta o disminueix, els valors obtinguts no seran del tot fiables. Els valors del magnetòmetre, tot i que l'error pot ser major que els de l'encoder, són més fiables però, per evitar aquests moments puntuals d'errors es fa una mitja entre els valors dels encoders i els del magnetòmetre.

CAPITOL 3. KINECT

3.1. Introducció

Kinect[1] (**Fig.3.1**) és un element desenvolupat per Microsoft que permet als usuaris controlar i interactuar amb la consola sense la necessitat de tenir contacte físic mitjançant una interfície natural d'usuari que reconeix moviment, objectes i imatges.

Inicialment va ser creat per ús exclusiu de la Xbox, però Microsoft se n'ha adonat del seu gran potencial.

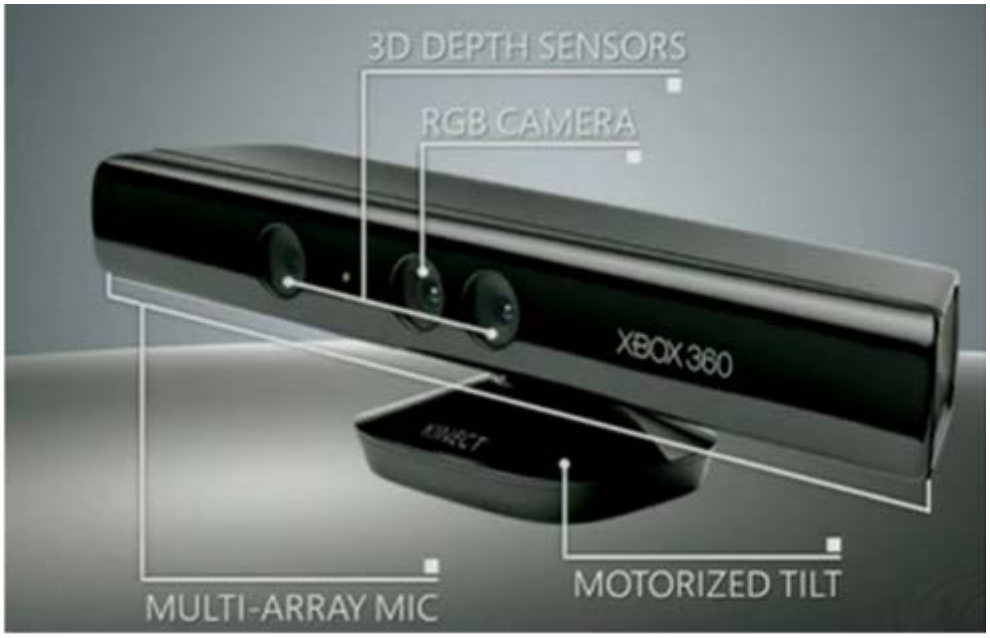


Fig. 3.1 Kinect

Compta amb una base giratòria motoritzada, tres sensors de moviment, una càmera de captació de moviment VGA amb una resolució de 640x480 píxels a 30FPS, i una doble càmera de profunditat de 3D de 640x480 píxels a 30FPS. A més compta amb 4 micròfons que poden reconèixer veus per separat.

El Kinect retorna dos tipus de imatges, la imatge de la càmera VGA, i la imatge de profunditat, proporcionada pel projector infraroig i la càmera infraroja.

La idea inicial va ser substituir el sensor central pel Kinect. La raó d'això és que els sensors proporcionen un únic punt d'informació cada vegada, i el Kinect retorna tots els punts que té en el seu camp de visió.

[1] http://www.xbox.com/es-ES/Kinect

D'aquesta manera, a partir del que retorni el Kinect, s'hauran d'extreure les distàncies, igual que es feia amb el sensor. Amb el sensor s'obtenia una distància cada vegada que retornava un valor, i amb aquest canvi, s'obtindran totes les distàncies desitjades que entrin dins de la imatge del Kinect.

En aquest cas, es treballarà amb la imatge de profunditat. Aquesta es basa en tons de grisos: el negre és el més a prop i el blanc el més lluny. S'ha de treballar sabent que el dispositiu té una sensibilitat a la llum i, per tant, els punts on influeixi la llum, els detectarà com a punts llunyans.

Després de profunditzar més en el camp de Kinect, es va veure que la idea inicial no era la més adequada. S'ha de tenir en compte que aquest model només detecta entre 80 cm i 4 metres. A partir d'aquestes distàncies, el dispositiu no retorna la posició. Per tant, aplicant el nostre cas, als 80 cm es deixaria de saber la distancia del robot, però es trobaria massa lluny per fer el gir. Així doncs, es decideix utilitzar el Kinect per a distàncies superiors a 80 cm i, a partir d'aquí es passaria a utilitzar com a referència de posició el sensor central.

3.2. Recol·lecció de dades

Per començar, es crearà un programa que el que farà serà mostra la imatge de profunditat i retornar la distància del punt que es cliqui en aquesta imatge.

El resultat que s'obté es mostra a la figura **Fig. 3.2**, on la distància del punt senyalat s'ha calculat a partir dels píxels X i Y de la imatge.

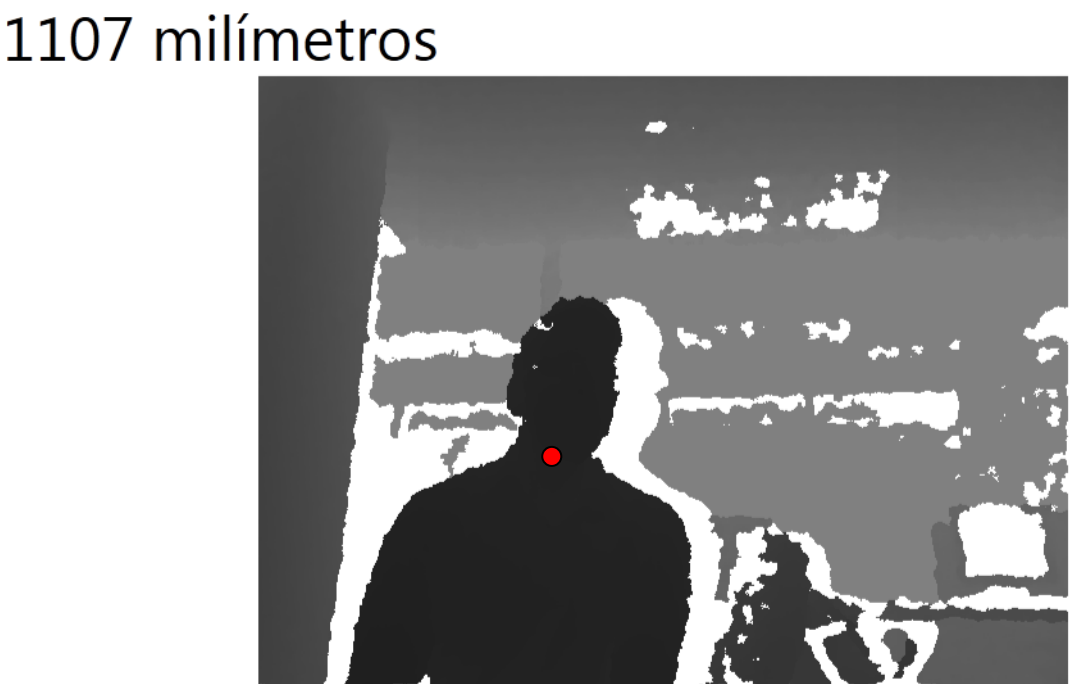


Fig. 3.2 Imatge de profunditat amb distancia d'un punt

Aquest programa s'anirà ampliant, per acabar obtenint els resultats desitjats.

A continuació, es farà que, en comptes de un punt, retorni les distàncies d'una línia de la imatge, la línia central. En aquest cas, el píxel Y es fixarà i serà constant i, el píxel X s'anirà incrementant fins a recórrer tota la línia.

En aquest cas, les distàncies seran guardades en un vector, i a continuació, plasmades en un gràfic, per tal de fer el resultat més visual i fàcil d'interpretar. En les figures **Fig. 3.3** i **Fig. 3.4,** es pot veure la imatge de profunditat que retorna el dispositiu amb una línia vermella orientativa de la part de la imatge amb la que es fan els càlculs i el gràfic resultant, respectivament.

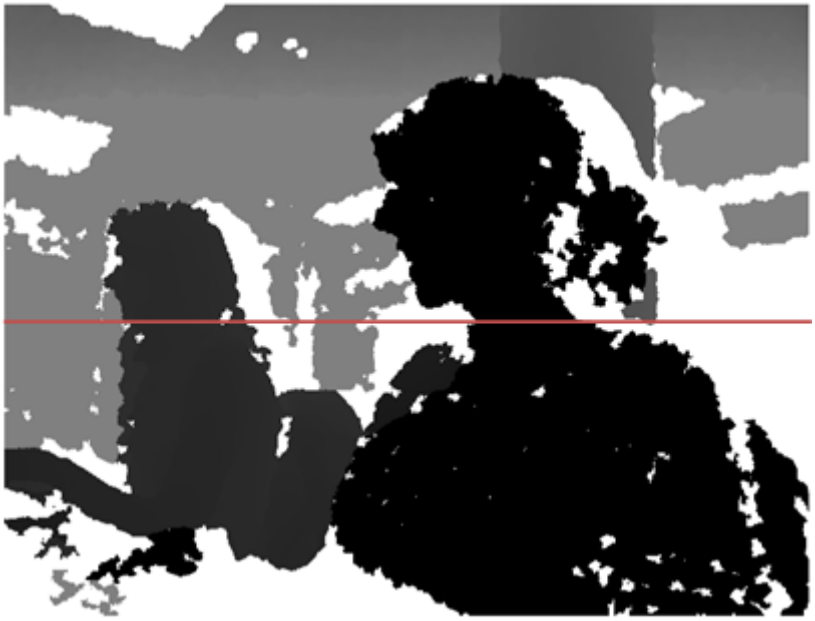


Fig. 3.3 Imatge de profunditat amb línia central

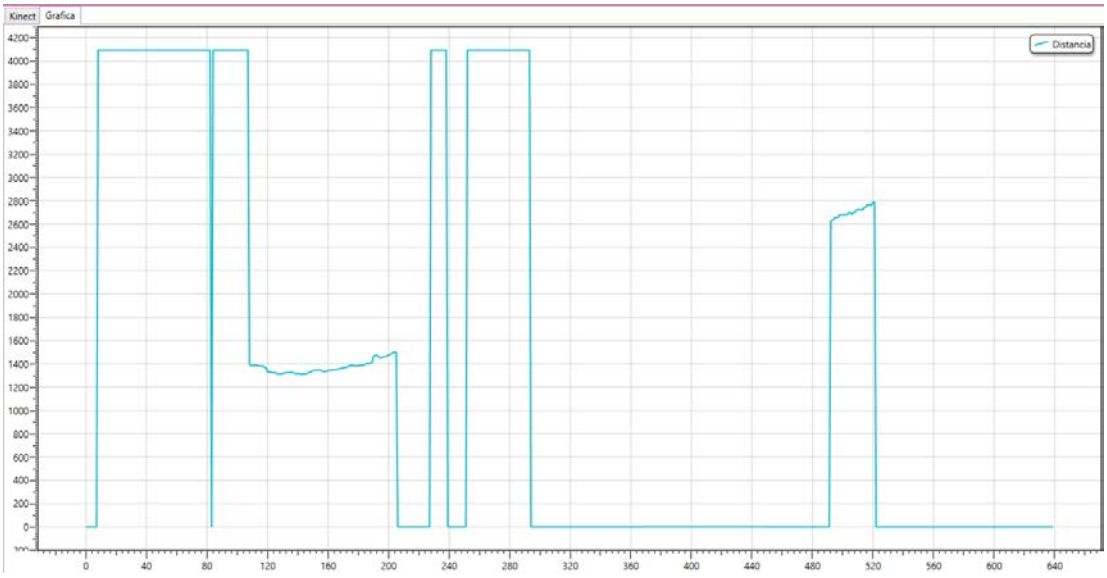


Fig. 3.4 Gràfic resultant amb les distàncies de la línia central

En el gràfic, es pot veure com la mínima distancia detectada són 80 cm. Per aquest motiu, quan hi ha una distancia que no pot detectar la fa 0. Això fa que hi hagi pics molt pronunciats i molts desnivells quan passa de detectar a no detectar i, al contrari.

Per evitar això, s'aplica a les distàncies obtingudes, un filtre de mitjana de finestra variable. La finestra utilitzada és de 20.
El que fa aquest filtre és agafar els valors de les posicions 0 fins la 19, ordenar els valors de petit a gran, agafar el valor central i col·locar-lo en un nou vector. A continuació, agafa els valors de les posicions 1 fins la 20 i fa el mateix procés anterior. Segueix amb el procés fins que arriba a la posició en la qual no té 20 per endavant i, per tant, no pot seguir amb el procés. Aquest nou vector, serà el que es plasmi en el gràfic.

A les imatges **Fig. 3.5** i **Fig. 3.6,** es pot veure la imatge de profunditat amb una línia vermella orientativa de la part de la imatge amb la que es fan els càlculs i el gràfic corresponent a aquesta imatge, respectivament. En el gràfic hi ha una línia blau clar, que mostra les distancies sense filtrar, i una línia blau fosc que mostra les distàncies un cop aplicat el filtre.

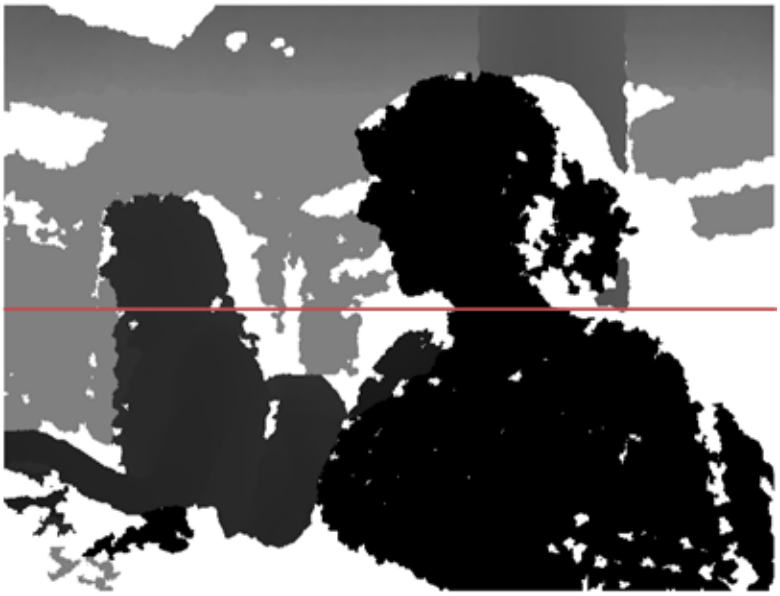


Fig. 3.5 Imatge de profunditat amb línia central

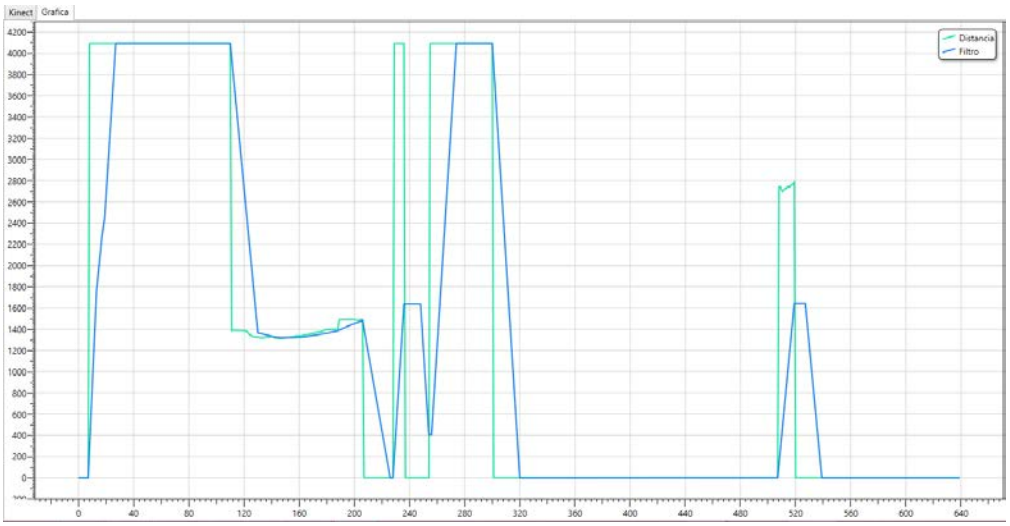


Fig. 3.6 Gràfic de distàncies amb filtre de mitjana

En el gràfic anterior es poden veure els canvis bruscos, com els comentats anteriorment, que al filtrar la distància, passen a ser variacions més atenuades.

Per continuar amb el procés, el següent pas és ampliar el número de punts que s'agafen per a calcular la distancia. El procediment és agafar els píxels de 5 línies de la imatge, la línia central, dos per amunt i dos per avall. El procés que es seguirà és: s'agafarà el primer píxel de cada línia, es farà la mitja i es guardarà en un vector. A continuació es repetirà el procés amb cada píxel, fins arribar a l'últim. Aquest vector serà el que contingui les distancies que es representaran en el gràfic.

A les imatges **Fig. 3.7** i **Fig. 3.8,** es pot veure la imatge de profunditat que mostra el dispositiu amb una línia vermella orientativa de la part de la imatge amb la que es fan els càlculs i, el gràfic resultant de fer la mitja de les 5 línies de la imatge, respectivament. En aquest cas, també s'observa en el gràfic una línia de color vermell on es veuen les distàncies sense filtrar, i una línia de color verd on es veuen les distàncies amb el filtre aplicat.

Fig. 3.7 Imatge de profunditat amb 5 línies centrals

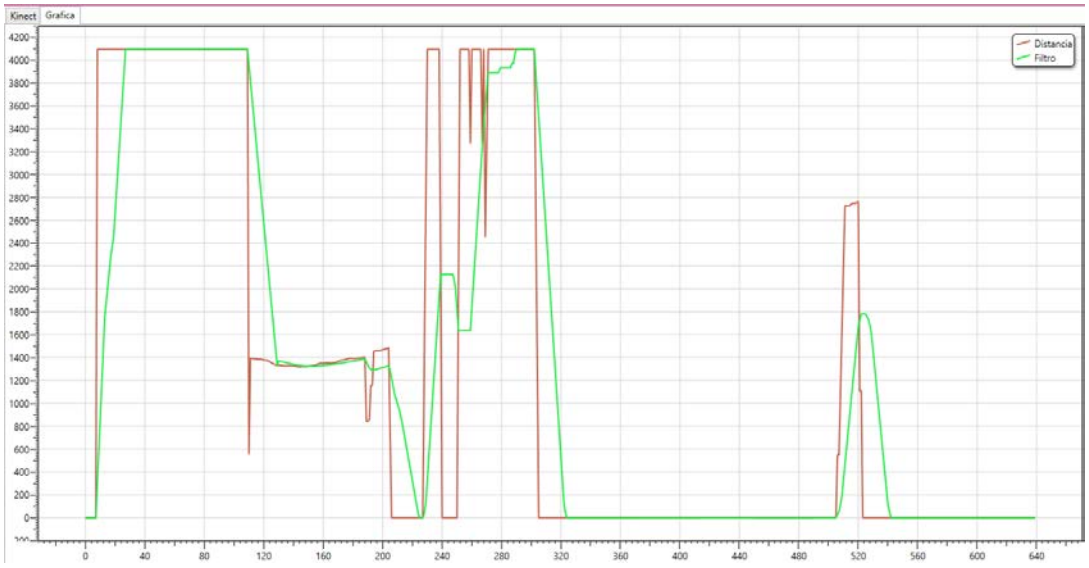


Fig. 3.8 Gràfic distàncies 5 línies amb filtre de mitjana

Un cop fetes les proves, aquest últim cas s'integra en el projecte. Per a poder fer-ho, es necessitat connectar el Kinect directament amb el cotxe. Per tant, serà necessària una placa extra on s'hi instal·li el sistema operatiu Windows, per processar les dades del Kinect.

3.3. Placa

La placa escollida és una EPIA n700[1]. S'escull aquesta, perquè és té prèviament al projecte.

Al mateix temps que es prepara la placa per al seu ús, s'ha de construir una plataforma per introduir-la a dalt del vehicle. Es realitzen diferents prototips que s'estudiaran més tard amb els seus avantatges i inconvenients. Aquests dos prototips es poden veure a la imatge **Fig. 3.9.** A la part esquerra apareix el primer prototip, fet amb fusta i plàstic i, a la dreta el segon, construït amb peces de Mecano.

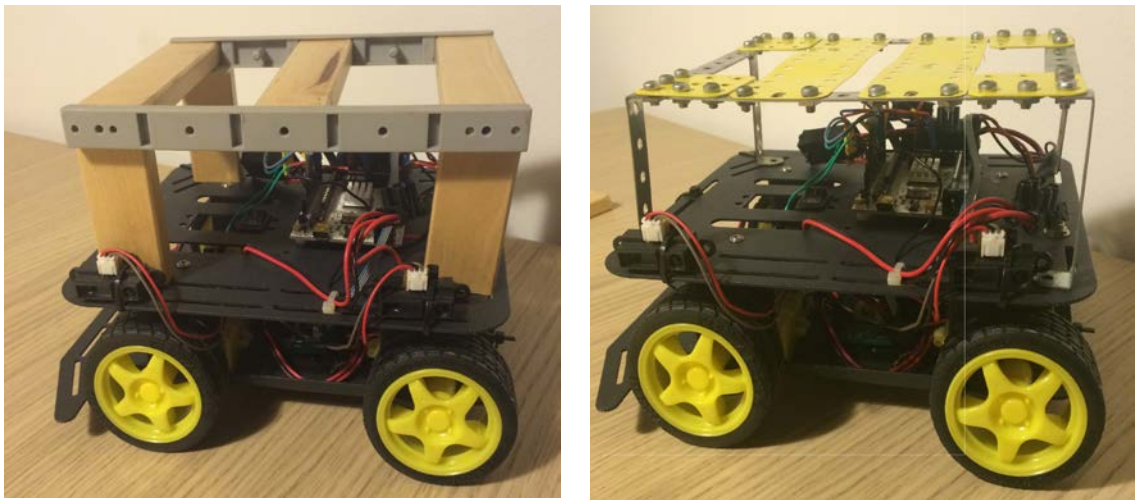


Fig. 3.9 Plataformes placa extra Kinect

3.4. Conclusions

Finalment, s'ha aconseguit extreure la informació del mapa de profunditat. A partir d'aquesta informació s'han pogut extreure els valors de la distància per a l'algorisme SLAM. Un cop s'han tingut tot aquest procés, s'ha escollit una placa adequada per a la incorporació del Kinect al projecte.
Per tema de pressupost no s'ha pogut adquirir la placa i la seva integració consta com a treball futur.

[1] http://www.viaembedded.com/en/products/boards/710/1/EPIA-N700_(EOL).html

CAPITOL 4.PID

4.1. Introducció

El PID (*Proportional Integral Derivative*) és un mecanisme de control per realimentació. Consisteix en calcular l'error que hi ha entre el valor que es vol obtenir i el valor mesurat amb l'objectiu d'obtenir un valor per disminuir aquesta diferència.

L'algoritme de càlcul de control de PID es divideix en tres paràmetre: el proporcional, l'integral i el derivatiu. El proporcional determina l'error estacionari actual, l'integral genera una correcció proporcional a l'error acumulatiu i, el derivatiu determina el temps en el que es produeix l'error.
Es representa en forma de funció de transferència:

(4.1)

On representa el guany proporcional, el guany integral i el guany derivatiu (veure [4] i [5]).

Ajustant aquestes tres constants en l'algoritme de control del PID es pot obtenir el valor de sortida desitjat. A la següent figura (**Fig. 4.1**) es descriuen les equacions i la relació entre elles.

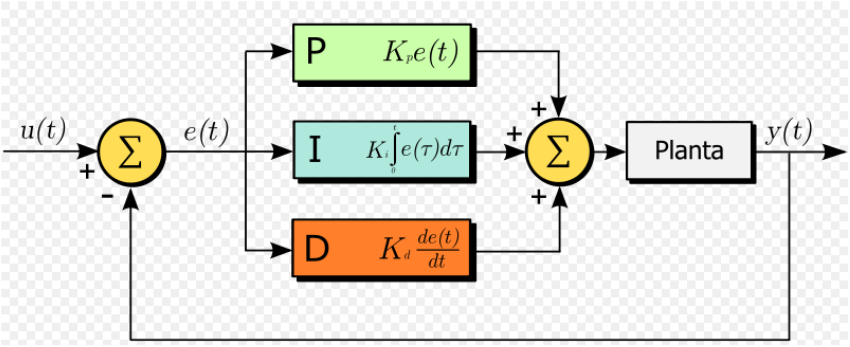


Fig. 4.1 Esquema funciona del PID

S'ha de tenir en compte que l'aplicació d'aquest algoritme no garanteix un control òptim del sistema ni la seva estabilitat. Algunes aplicacions poden requerir l'ús d'una o dues accions. El controlador PI és bastant comú ja que l'acció derivada (D) és molt sensible al soroll de mesura.
En aquest projecte s'ha aplicat un controlador PI adaptat a les necessitats. A continuació, s'explicaran les dues adaptacions del controlador en dos casos diferents.

4.2. Seguiment de paret

El primer cas que s'explicarà és l'algoritme adaptat per la realització del seguiment de paret.

En aquest cas concret, l'objectiu d'aquest algoritme és corregir els errors introduïts pel propi moviment del vehicle per aconseguir que aquest vagi paral·lel a la paret. S'ha de trobar una fórmula que corregeixi aquest error evitant que hi hagi moltes correccions per a que el moviment sigui continuat.

El procediment que s'explicarà a continuació es durà a terme sempre i quan el sensor central sigui superior al valor mínim determinat per fer el gir. Aquest valor es determinarà més endavant quan s'introdueixi l'altre algoritme.

Primer s'han de marcar els límits, és a dir, la distància màxima i la distància mínima a la paret. La distància màxima està determinada pels sensors. Aquests sensors només obtenen valors a distàncies menors de 30 centímetres. Per tant, es defineix la distancia màxima a 25 centímetres.

La distància mínima ve determinada per l'espai que necessita el vehicle per poder realitzar el gir i pels components extres afegits. El Kinect sobresurt de l'ample de vehicle. Inicialment es realitzaran els càlculs amb un valor determinat i quan s'afegeixi el Kinect es modificarà. De moment, la distància mínima haurà de ser més de la meitat de l'ample del rover.

Això vol dir que el vehicle té 15 centímetres de marge per moure's. La situació òptima seria que el vehicle sempre es mantingués a 17,5 centímetres de la paret però cal recordar que el gir farà variar aquesta distància i, sense marge d'error, hi hauria masses correccions.

Un cop definits els límits, es determinarà els diferents casos que es poden donar segons la magnitud de l'error. L'error és la diferència absoluta entre el dos sensors laterals i es calcularà a partir dels valor obtinguts d'aquests mateixos. S'ha de marcar un valor per identificar quan s'han de realitzar les correccions a partir de l'error descrit anteriorment. Es distingiran dos casos: quan l'error sigui més gran que el valor establert i quan sigui més petit.

Quan l'error és superior al valor (**Fig. 4.2**) es suposarà que hi ha una cantonada i es realitzarà un gir a l' esquerra.

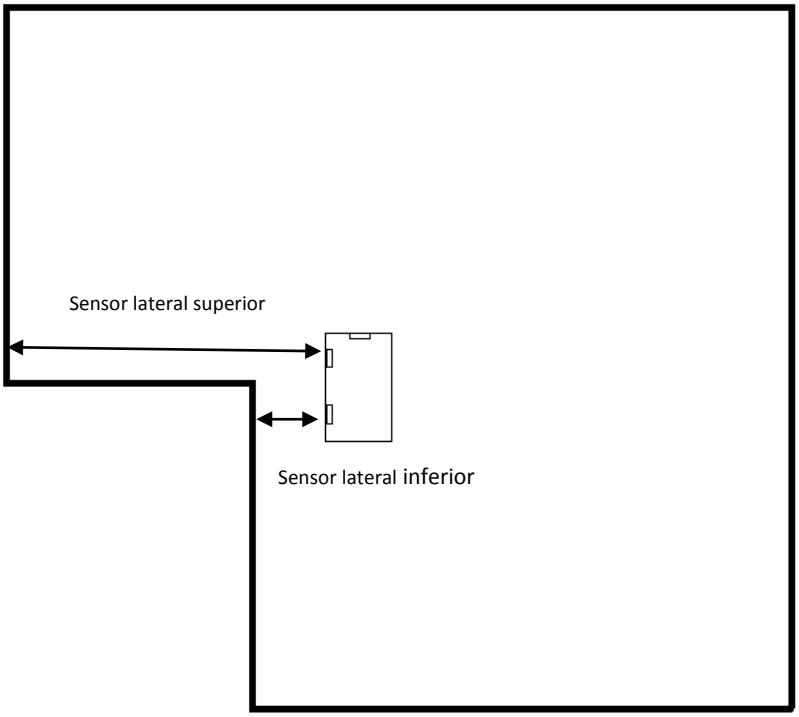


Fig. 4.2 Cas 1. Gir a l'esquerra

Quan l'error és inferior al valor s'haurà de distingir entre els dos possibles casos:

- Valor del sensor lateral superior més gran que el del sensor lateral inferior, és a dir, el cotxe s'està allunyant de la paret (**Fig. 4.3**). En aquest cas, s'haurà de fer una modificació girant cap a l'esquerra.

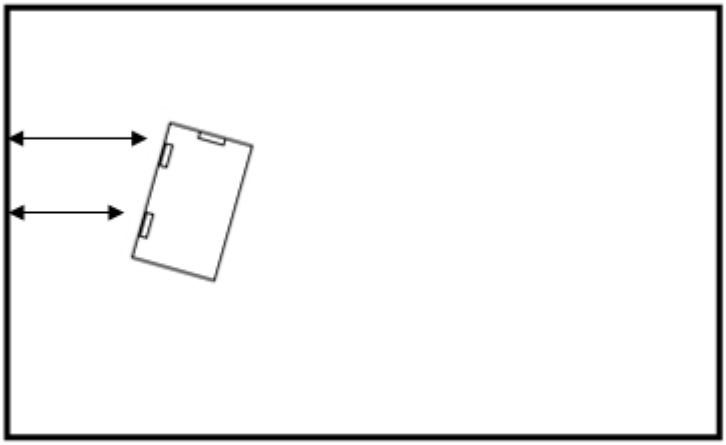


Fig. 4.3 Cas 1. Correcció allunyament paret

- Valor del sensor lateral superior més petit que el del sensor lateral inferior, és a dir, el cotxe s'apropa a la paret (**Fig. 4.4**). En aquest cas, l'error es modificarà girant cap a la dreta.

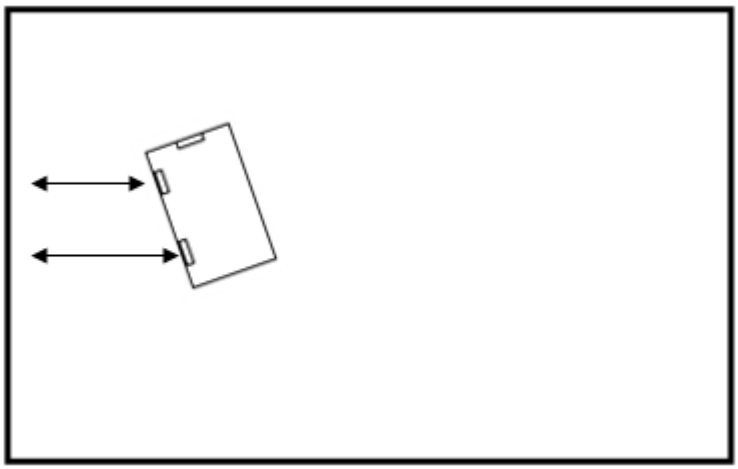


Fig. 4.4 Cas 2. Correcció apropament paret

A partir d'aquesta limitació s'hi ha d'afegir una de nova, fer que l'error tingui un valor mínim concret, per evitar que es realitzin correccions petites, i aconseguir un moviment fluït.

Finalment, un cop es coneix el cas en que es troba el vehicle, s'ha de calcular l'angle de gir depenent de l'error (diferència de sensors) calculat prèviament.

Els valors coneguts són la longitud del vehicle i les distàncies en mil·límetres dels sensors laterals amb els quals, com s'ha dit anteriorment, s'extreu la diferència. A la **Fig. 4.5** es visualitzen els paràmetres esmentats.

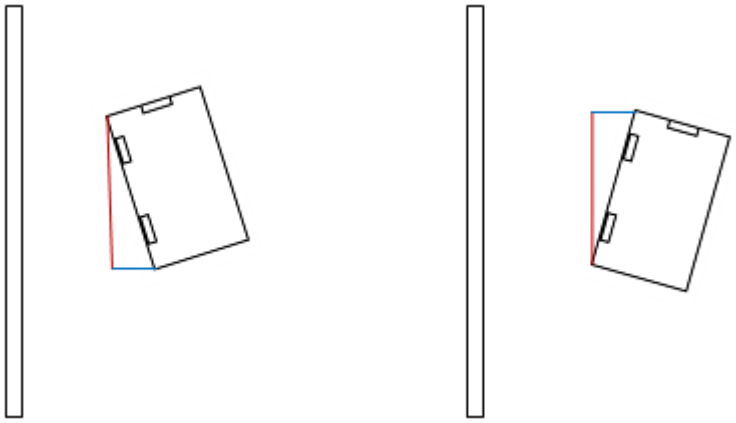


Fig. 4.5 Paràmetres coneguts: vermell longitud rover, blau diferencia sensors

Tal i com es pot deduir amb la imatge, la longitud del vehicle equival a la hipotenusa d'un triangle i, la diferència entre els sensors al catet oposat de l'angle a calcular. Tot això es pot veure resumit a la següent fórmula:

$$\alpha = arcsin\left(\frac{(S-s)*10}{longitud\ rover}\right) \quad (\mathbf{4.2})$$

On **S** és el sensor de major valor, **s** el sensor de menor valor i **α** l'angle de gir.

Com que els valors que ens retorna els sensor estan en centímetres i la longitud del vehicle està en mil·límetres, s'ha de multiplicar per un factor 10 el dividend.

Aquest angle no ha de ser mai superior a 1 per tal d'evitar un efecte contrari al desitjat, que és obtindre un moviment fluït.

4.3. Gir

El segon cas que s'explicarà és l'algoritme adaptat per la realització del gir a la dreta.

L'objectiu d'aquest algoritme és girar gradualment, dit d'una altra manera, realitzar girs petits i constants fins que el vehicle es col·loqui a la posició desitjada.

Com a referència s'utilitzarà el valor obtingut del sensor central. Durant el gir s'haurà de calcular el cosinus d'aquest valor per tal d'obtenir la distància real a la paret.

Primerament s'ha d'establir a partir de quina distància es començarà a fer el gir amb PID i la distància límit a partir de la qual el gir es farà sense PID. Amb aquest últim paràmetre s'evita que el vehicle xoqui ja que la informació rebuda dels sensors s'actualitza cada 150 mil·lisegons.

La fórmula generalitzada que relaciona la distància i els graus de gir és:

$$\alpha = \frac{distancia\ a\ la\ paret}{distancia\ a\ la\ paret - mínima\ distancia\ a\ la\ paret} * K \quad (\mathbf{4.3})$$

on *distancia a la paret* es calcula prèviament amb els paràmetres establerts i *K* és un factor de multiplicació.

Distància a la paret es defineix de la següent manera:

$$distància\ a\ la\ paret = valor\ sensor\ central * \cos(angle\ de\ gir\ total) \quad \textbf{(4.4)}$$

L'angle de gir total es calcula de la següent manera:

$$angle\ de\ gir\ total = angle\ relatiu - \ angle\ inicial \quad \textbf{(4.5)}$$

on *l'angle relatiu* és la posició obtinguda del magnetòmetre en aquell moment i *angle inicial* és la posició del gir anterior.

A continuació, es definirà el procediment que es seguirà per realitzar el càlcul de gir després del primer gir.

Primer s'haurà de comprovar que la diferència entre els sensors laterals és major que la histèresis definida (histèresis igual a 1). Això voldrà dir que el vehicle no està alineat a la paret i, per tant, s'ha de continuar girant.

Seguidament, s'actualitza el valor del sensor central i l'angle relatiu per així tornar a calcular la variable angle de gir total i es modifica la variable angle inicial.

Després de tot això ja es pot utilitzar la fórmula de la distància a la paret amb els valors correctes.

Aquest procediment es realitzarà tants cops com sigui necessari fins que el vehicle estigui paral·lel a la paret.

4.4. Implementació

Amb la finalitat d'obtenir uns resultats positius, s'implementaran els algoritmes d'un en un. Un cop introduïts els dos algoritmes es realitzaran les modificacions necessàries per aconseguir un bon funcionament. El més senzill de tots dos és el PI de seguiment de paret.

S'introdueix l'algoritme a la funció Run() de la classe Engine.cs. Aquesta funció posa en marxa el vehicle i el mou mentre no succeeixi cap esdeveniment que el cancel·li. Es distingeixen dos casos: buscant paret i seguiment de paret.

El cas buscant paret s'utilitza al inici del moviment. El vehicle anirà en línia recta fins trobar una paret moment en el qual, canviarà de cas.

El cas seguiment de paret és el cas que s'utilitza durant tot el recorregut. Amb els sensors laterals va identificant la distancia a la que es troben els obstacles i amb el sensor frontal controla la distancia dels obstacles frontals per tal de realitzar els girs i evitar xocar-se.

A la imatge **Fig. 4.6**., es mostra el resultat que s'espera que tingui el rover després de localitzar la paret. Ha de continuar tot recte i procurar mantenir la distància amb la paret de la part esquerra. Si no és així realitzar les correccions mínimes necessàries.

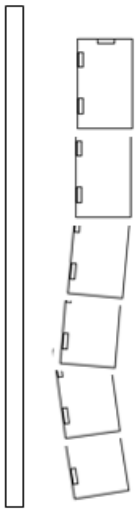


Fig. 4.6 Seguiment de paret esperat

Quan la distància donada pel sensor central és igual o menor dels 40 centímetres, es procedirà a realitzar el gir.

El procediment que segueix és el següent: es realitza un primer gir i després es recalcula l'angle de gir depenent de la posició del vehicle fins col·locar-se paral·lel a la paret.

El primer gir serà aproximadament un terç del gir total a realitzar per tal de que el moviment sigui continu.

Per finalitzar aquest apartat, es realitza una prova amb els dos PI implementats (**Fig. 4.7**).

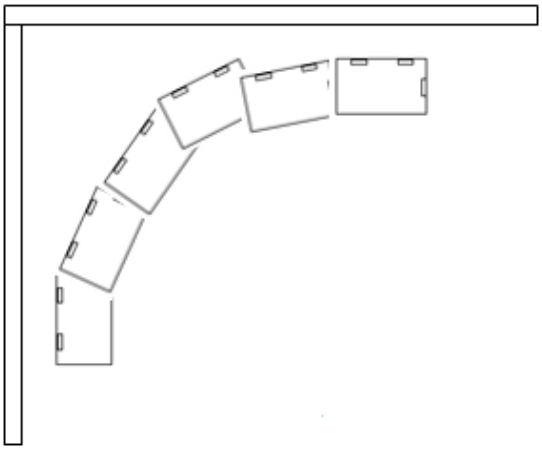


Fig. 4.7 Gir esperat

4.5. Conclusions

PID és un algoritme senzill conceptualment parlant i de fàcil implementació amb la capacitat de l'adaptació segons el problema que es presenti. S'ha establert un nombre de correccions menors, per evitar que el robot tingui un moviment menys fluït. Al PI s'han implementat els girs cap a la dreta.

CAPITOL 5. CONNECTIVITAT

5.1. Introducció

En aquest capítol s'explicaran els dos grans blocs de comunicació del projecte: XBee i Bluetooth. Generalitzant el projecte es resumeix en tres gran blocs individuals (**Fig.5.1**).

Fig. 5.1 Esquema general de connectivitat

Anteriorment, es tenia la comunicació entre l'estació base i el rover, mitjançant uns mòduls XBee. Aquests mòduls es substitueixen per uns altres i es configuren de nou.

Per altre banda, s'ha d'afegir una nova comunicació bluetooth, entre el dispositiu mòbil i el cotxe, ja que es crea una aplicació mòbil per moure remotament el cotxe, que s'explicarà mes endavant.

La comunicació entre el dispositiu i el vehicle no serà directe ja que el ports sèries de la placa ja estan ocupats pels mòduls XBee. Per tant, s'utilitzarà l'estació base com a element pont entre l'aplicació i el rover (**Fig. 5.2**)

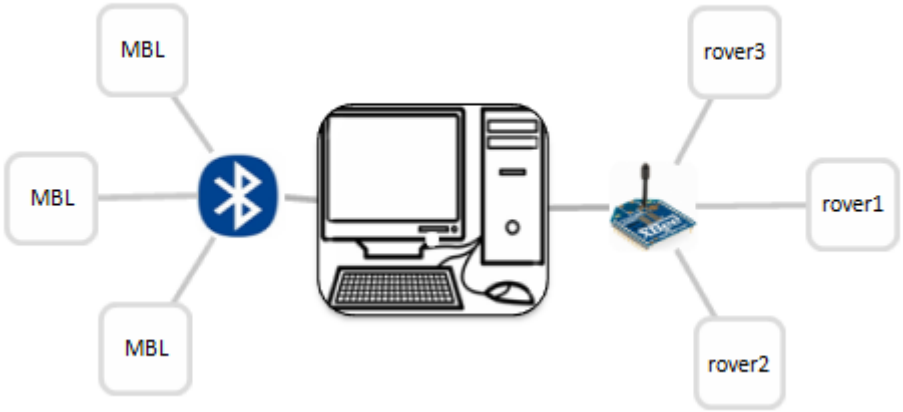


Fig. 5.2 Esquema de comunicació

D'aquesta manera, s'utilitzarà una comunicació bluetooth entre el dispositiu i l'estació base i, una comunicació zigbee (programada anteriorment) entre aquesta i el vehicle.

5.1. XBee

A continuació s'explicarà el que és un mòdul XBee, i la configuració que s'ha fet en cada un dels mòduls utilitzats en el projecte.

5.1.1. Introducció

Els mòduls XBee són dispositius que integren un transmissor-receptor de ZigBee i un processador en un mateix mòdul.

Fig. 5.3 Mòdul XBee

ZigBee (veure [5]) és un protocol de comunicació sense fils basat en el estàndard de comunicacions per a xarxes sense fils IEEE_802.15.4. És desenvolupat per ZigBee Alliance formada per centenars de companyies que volen solventar la necessitat d'un estàndard per a comunicacions a baixa velocitat, amb un baix cost de implementació i on els dispositius que formen part de la xarxa poden requerir baix consum.

Les característiques dels dispositius ZigBee inclouen:

- Velocitat de transmissió entre 25 i 250 kbps.
- Protocol asíncron, half-duplex i estandarditzat.
- Múltiples topologies de xarxa com punt a punt, punt a multipunt o xarxa en malla.
- Les comunicacions es realitzen en la banda lliure de 2,4 GHz.
- És un protocol segur, ja que es pot implementar encriptació i autentificació

En les xarxes ZigBee es troben tres tipus de dispositius:

- *Coordinator*. En totes les xarxes hi ha un coordinador, només un per xarxa. Entre les seves tasques estan les de formar i gestionar la xarxa.

- *Router*. Són dispositius de la xarxa que tenen la capacitat d'enviar i rebre informació. Poden actuar com missatgers entre dispositius que estan molt allunyats per establir una comunicació directament. Els dispositius ZigBee estan pensats per treballar en xarxes i cobrir llargues distancies passant la informació entre els diferents nodes.
- *End device*. Aquests serien els dispositius de baix consum. Poden enviar i rebre informació però no poden actuar com a missatgers entre altres dispositius de la xarxa. Els end-device solen estar en un mode de baix consum i es desperten quan volen enviar o rebre informació. Necessiten estar associats a un coordinador o router, que guardin els missatges que han sigut enviats per ells mentre estaven dormits i els hi facin arribar quan despertin.

Algunes de les principals característiques dels mòduls XBee (veure [6]) són:

- Abast: fins a 100 metres en línia de visió.
- 9 entrades/sortides amb entrades analògiques i digitals.
- Baix consum <50mA quan estan en funcionament i <10µA en mode sleep.
- Interfície serial.
- Fàcils d'integrar.

Existeixen 2 series d'aquests mòduls. La sèrie 1 i la sèrie 2, també coneguda com 2.5. Els mòduls de la sèrie 1 i la sèrie 2 tenen el mateix pin-out, no obstant, no són compatibles entre si ja que utilitzen diferents chipset i treballen amb protocols diferents.

La sèrie 1 està basada en el chipset Freescale i està pensada per ser utilitzada en xarxes punt a punt i punt a multipunt. Els mòduls de la sèrie 2 estan basats en el chipset de Ember i estan dissenyats per ser utilitzats en aplicacions que requereixen repetidors o una xarxa mallada. Ambdós mòduls poden ser utilitzats en els modes AT i API, explicats a continuació.

- Mode AT. El que el mòdul rep pel pin DIN, ho envia per la comunicació sense fils tal qual. I de la mateixa manera, el que el mòdul rep ho transmet al microcontrolador com li arribar pel pin DOUT. Si enviem +++, el mòdul entra en el mode comanes i ens permet configurar diferents opcions a través del seu port sèrie.
- Mode API. La comunicació amb el mòdul és més complexa. Ja no entra i surt el que enviem al mòdul pel port sèrie, sinó que ens comuniquem amb el mòdul XBee mitjançant *frames*, és a dir, les dades han d'anar estructurades segons un ordre establert. Amb aquesta opció de comunicació, tenim moltes més possibilitats de comunicació (podem enviar comanes a altres mòduls per configurar-los remotament, saber qui es el remitent del missatge...).

Cada mòdul té assignat un número de sèrie. Aquest numero es pot trobar darrere del mòdul XBee. Esta format per 64 bits. Els primers 32 bits corresponen amb un numero assignat pel fabricant, i els següents 32 bits són

individuals per a cada mòdul, no hi haurà dos mòduls amb aquest numero igual. Aquest numero es pot utilitzar per identificar un mòdul dins d'una xarxa i comunicar-se amb ell.

En aquest cas, es disposa de dos mòduls de Digi: XBee Serie 2[1]. Per poder integrar els mòduls en el projecte, es necessita un XBee Explorer USB (**Fig. 5.2**).

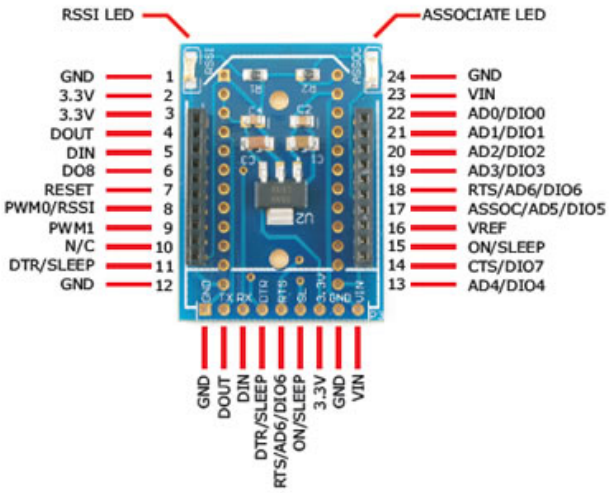


Fig. 5.4 XBee Explorer USB

Aquesta placa permetrà comunicar l'ordinador amb el mòdul XBee. És un xip que fa de pont entre el USB del ordinador i la UART del microcontrolador. S'utilitzarà per actualitzar, descarregar firmware i configurar els mòduls XBee. També s'utilitzarà per a dotar al ordinador de connexió ZigBee i connectar-lo a la xarxa de dispositius per enviar i rebre dades.

5.1.2. Configuració

Per canviar el firmware i configurar els mòduls s'utilitzarà un programa, X-CTU.

Es configurarà un dels mòduls com a *Coordinator AT* i l'altre mòdul com a *Router AT*. Primer s'hauran de configurar les característiques del mòdul, i a continuació els paràmetres de la comunicació.

Els paràmetres que s'han de configurar són:

- Function Set
- Modem XBEE
- Versió

[1] ftp://ftp1.digi.com/support/documentation/90000866_A.pdf

- PAN ID. Numero d'identificació de la xarxa. Tots els dispositius de la mateixa xarxa han de tenir el mateix PAN ID.
- Destinations Address High: Primers 32 bits del numero de sèrie del mòdul amb el que es vol comunicar
- Destination Address Low: Últims 32 bits del numero de sèrie del mòdul amb el que es vol comunicar

A continuació es mostra la configuració que s'estableix en cada mòdul (**Taula 5.1**).

Taula 5.1 Configuració de cada mòdul

Function Set	Modem XBee	Version	PAN ID	Destination Address High	Destination Address Low
Coordinator AT	XXB24-ZB	Ultima	6666	0013A200	4081A2AD
Router AT	XX24-ZB	Ultima	6666	0013A200	4081A209

Un cop configurats els mòduls, es comprova que hi hagi comunicació entre ells (**Fig. 5.5**) amb el propi programa Es connecta el mòdul configurat com a Coordinator al ordinador, que estarà reben i emeten dades. Els caràcters que es teclegen apareixen en blau, i són els que s'estan enviant a l'altre mòdul. El que ens interessa és rebre dades des de l'altre mòdul. El que rebem apareixerà a la pantalla de color vermell. Un cop connectats els dos mòduls es comprova la connexió.

Fig. 5.5 Comunicació mòduls XBee

5.2. Bluetooth

Bluetooth (veure [7]) és una especificació tecnològica per a xarxes sense fils d'àrea personal (WPAN) que permet la transmissió de veu i dades entre diversos dispositius mitjançant una radiofreqüència segura (2.4GHz). Aquesta tecnologia ens facilita la comunicació entre dispositius mòbils i fixes, l'eliminació de cables i connectors i, ofereix la possibilitat de crear petites xarxes per tal de sincronitzar dades entre equips.

Els dispositius que utilitzen habitualment aquesta tecnologia són del sector de les telecomunicacions i la informàtica personal (PDA, telèfons mòbils, ordinadors, impressores, càmeres digitals...)

El Bluetooth es poden classificar segons la classe a la que pertanyen (**Taula 5.2**) i segons l'ample de banda (**Taula 5.3**).

Taula 5.2 Classes de bluetooth

Classe	Potència Màxima Permesa (mW)	Potència Màxima Permesa (dBm)	Abast (aproximat)
Classe 1	100 mW	20 dBm	~ 30 metres
Classe 2	2,5 mW	4dBm	~ 10 - 5 metres
Classe 3	1 mW	0 dBm	~ 1 metre

Taula 5.3 Classificació segons ample de banda

Versió	Ample de Banda
Versió 1.2	1 Mbit/s
Versió 2.0 + EDR	3 Mbit/s
Versió 3.0 + HS	24Mbit/s
Versió 4.0	24Mbit/s

A continuació, es resumirà les característiques principals de les versions anomenades anteriorment:

- **Versió 1.2** → Permet la connexió i detecció de dispositius ràpidament. Utilitza la tècnica AFH, salts de freqüència adaptable d'espectre ampliat, que millora la resistència a les interferències de radiofreqüència. Velocitat de transmissió de 721 Kbits/s. La V1.2 utilitza connexions síncrones esteses (ESCO) que millores la qualitat de la veu dels enllaços àudio al permetre la retransmissió de paquets corruptes. Va introduir el control de flux i els modes de retransmissió L2CAP[1].

[1] Protocol de control i adaptació del enllaç lògic (L2CAP) que suporta la multiplexació de més alt nivell de protocol, segmentació i flux a través del control de flux i els modes de retransmissió.

- **Versió 2.0 + EDR** → La principal diferència és l'augment de la velocitat de transmissió de dades (EDR "*Enchanded Data Rate*"). La tassa nominal d' EDR és de 3Mbits/s però a la pràctica és de 2,1Mbits/s. EDR utilitza una combinació de GFSK (*Gaussian Frequency-Shift Keying*) y PSK (*Phase-Shift Keying*) amb dues variants, π/4-DQPSK i 8DPSK. EDR pot proporcionar un menor consum d'energia gràcies a un cicle de treball reduït.
- **Versió 3.0 + HS** → Suporta velocitats de transmissió de dades de fins 24Mbits/s, encara que no amb enllaços Bluetooth pròpiament dit. Aquesta versió utilitza la connexió bluetooth nativa per a la navegació i, la transmissió de dades a alta velocitat ho fa mitjançant un enllaç 802.11. La principal novetat és l'AMP (*Alternate MAC/PHY*) que permet l'ús d'alternatives MAC y PHY per al transport de dades de perfil Bluetooth.
- **Versió 4.0** → aquesta última versió inclou totes les característiques d'un Bluetooth clàssic més alta velocitat i protocols de baix consum. Té un abast de més de 100 metres, un reforçament de la seguretat, protecció de dades sota un xifrat de 128 bits, una nova tecnologia de baix consum energètic BLE (*Bluetooth Low energy*) i pot arribar a velocitats màximes de 1Mbit/s en la capa física.

Resumint, les especificacions tècniques de Bluetooth defineixen un canal de comunicació a un màxim de 720Kbits/s amb una distància òptima de 10metres.

El Bluetooth utilitza la tècnica FHSS (Espectre eixamplat per salts de freqüència) que consisteix en dividir la banda de freqüència en 79 canals, denominats salts, d'1MHz d'ample cadascú i, després, transmetre el senyal usant una seqüència de canals que serà coneguda per l'estació emissora com per la receptora.

Per tant, al canviar de canals amb una freqüència de 1600 vegades per segon, el Bluetooth pot evitar la interferència amb altres senyals de ràdio.

Per aconseguir l'objectiu de baix consum i baix cost es va idear una solució que es pot implementar en un únic xip utilitzant circuits *CMOS*[1].

Finalment, després de fer un petit estudi de les diferents característiques de les diverses versions de bluetooth existents i de les necessitats del projecte, el dispositiu empleat per a la comunicació serà un mini bluetooth v4.0 usb adapter.

Les característiques específiques d'aquest dispositius són: rang de 30 peus (9,14m aproximadament), tecnologia BLE i fàcil paritat amb altres dispositius sense fils.
L' estàndard Bluetooth es basa en el mode d'operacions mestre/esclau amb un màxim de 10 esclaus per xarxa. Un dispositiu mestre, o servidor, pot enllaçar-se amb 7 esclaus actius, clients, però només amb un pot mantenir la

[1] Família lògica utilitzades en la fabricació de circuits integrats. Aquesta tecnologia inclou microprocessadors , memòries, processadors digitals de senyals. La característica principal és el seu baix consum.

comunicació. La resta es troben a l'espera. A la **Fig. 5.6** es mostra la xarxa de l'estàndard de comunicació.

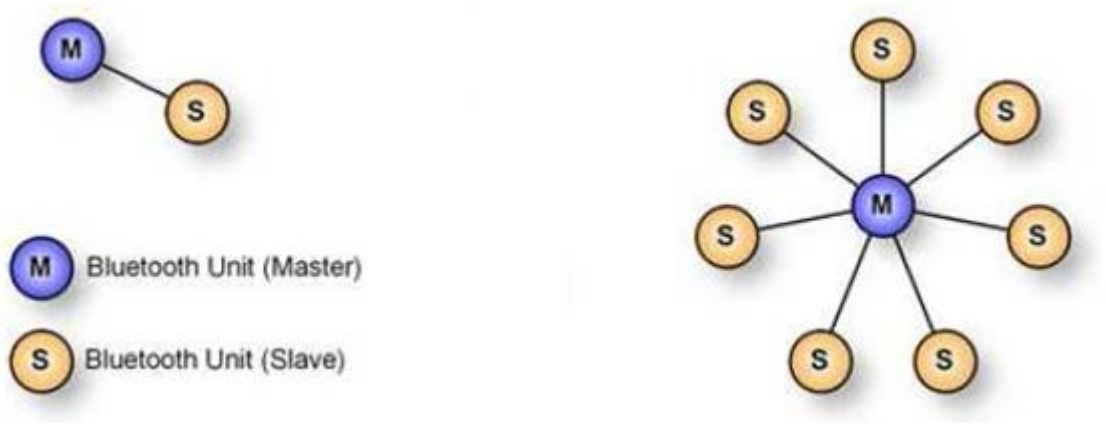


Fig. 5.6 Esquema xarxa de comunicació

En aquest projecte només hi haurà un servidor i un client però si en un futur es vol ampliar, el sistema estarà preparat. La comunicació es realitza mitjançant dues classes (**Fig. 5.7**).

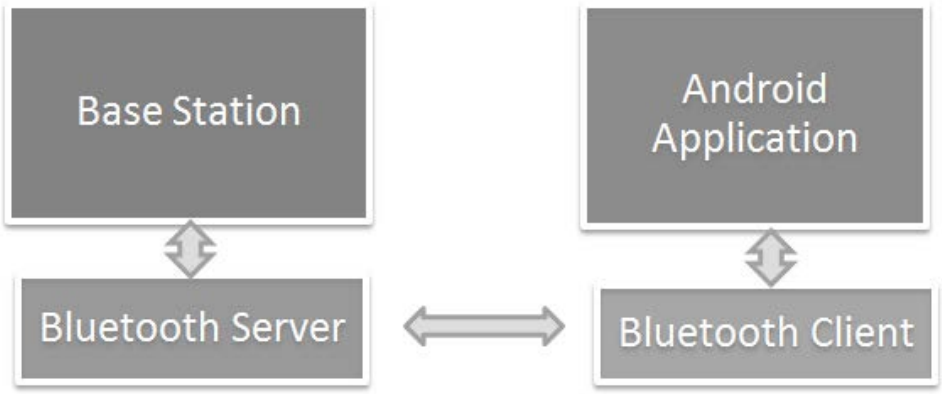


Fig. 5.7 Esquema comunicació bluetooth

S'explicarà aquest esquema amb profunditat en el capítol de control remot.

5.3. Conclusions

Després de fer un estudi exhaustiu de les diferents connexions de comunicació, s'arriba a la conclusió de que els dos dispositius tenen l'opció de crear xarxes amb l'objectiu d'ampliar l'escenari.

En el cas del mòduls XBee, només s'utilitzen dos (com una comunicació punt a punt), però s'hi podrien afegir més, formant així, una xarxa en malla. D'aquesta manera tots els mòduls es podrien comunicar entre ells. Per exemple, es podrien controlar diferents robots des de la mateixa estació base.

En el cas de la comunicació bluetooth, s'utilitzen dos dispositius: un mestre i un esclau. Aquesta xarxa es podria ampliar amb un màxim de 10 esclaus.

CAPITOL 6. CONTROL REMOT

6.1. Introducció

En aquest capítol es descriuran els dos dispositius afegits per al control remot del vehicle. Aquestes incorporacions permetran al usuari tenir un control més directe del rover.

Existiran tres maneres de controlar remotament el robot. La primera d'elles és des de l'estació base, part implementada en la primera versió del projecte. Les altres dues són les que es duran a terme en aquest projecte: des d'un dispositiu mòbil i des de un dispositiu anomenat Leap Motion.

Com s'ha explicat anteriorment, la comunicació entre el mòbil i el rover, passarà per l'estació base i, aquesta s'encarregarà d'enviar-li la instrucció al rover. De la mateixa manera, el dispositiu Leap es comunicarà amb l'estació base i aquesta, amb el rover (**Fig. 6.1**).

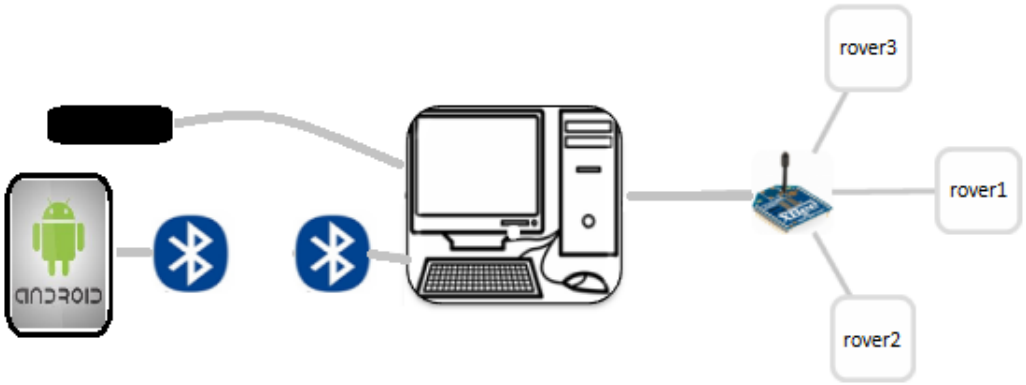


Fig. 6.1 Comunicació control remot

Per poder implementar aquests nous mètodes, és necessària la introducció d'una classe que processarà els paquets rebuts i depenent del paràmetre obtingut, es cridarà la funció del moviment corresponent, definida en la primera versió del projecte.

A continuació s'explicaran amb detall els dos nous mètodes aplicats.

6.2. Aplicació mòbil

6.2.1. Introducció

Per tal d'oferir a l'usuari una altra alternativa per moure el vehicle, es realitzarà una aplicació mòbil programada en java amb l'ajuda de l'Eclipse (veure[10]). Aquesta aplicació només podrà ser utilitzada amb dispositius que tinguin el sistema operatiu Android[1].

L'objectiu d'aquesta aplicació és poder connectar el dispositiu mòbil amb l'estació base amb la finalitat d'enviar ordres al robot.

L'aplicació mòbil tindrà dues parts ben diferenciades: una Interfície visual i una comunicació, en aquest cas, via bluetooth. Primer es realitzarà una interfície base que inclourà les funciones bàsiques i, un cop acabada la primera part, s'iniciarà la programació de la comunicació.

La interfície bàsica constarà (**Fig. 6.1**) de quatre botons per moure el vehicle endavant, endarrere, a esquerra i dreta, una barra per controlar la velocitat i, finalment, un botó per poder sortir de la aplicació.

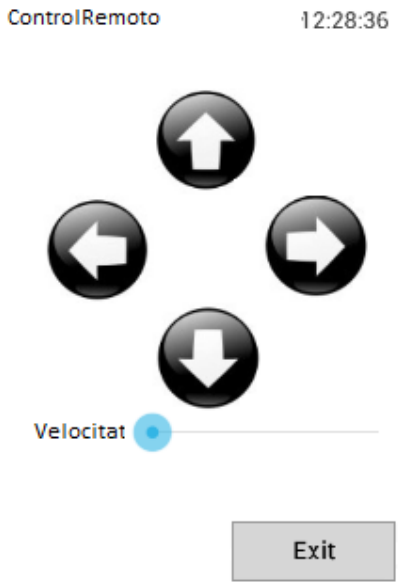


Fig. 6.2 Front-end app

Habitualment, els telèfons mòbils tenen activada la propietat de rotació de pantalla. Android té una funció encarregada de detectar la rotació i automàticament canvia la vista per ajustar-lo a les noves necessitats. Prèviament s'ha de crear una nova vista amb la distribució desitjada dels elements a mostrar (**Fig. 6.2**).

[1] http://developer.android.com/guide/index.html

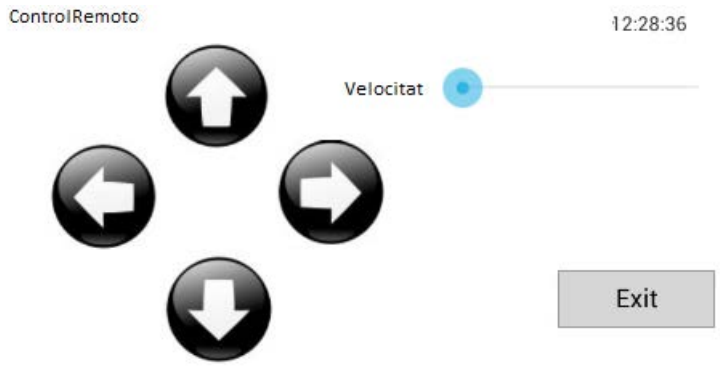


Fig. 6.3 Layout-land app

6.2.2. Implementació

Aquest apartat està enfocat a l'explicació de cóm es realitza la comunicació i els passos a seguir.

La comunicació entre ambdós dispositius seguirà els següents passos (veure[9]):

1.- Comprovació si el bluetooth del dispositiu mòbil està encès.
2.- Cerca dels dispositius.
3.- Connexió a l'estació Base.
4.- Inicialització de la comunicació.
5.- Transmissió de dades.
6.- Finalització de la comunicació.

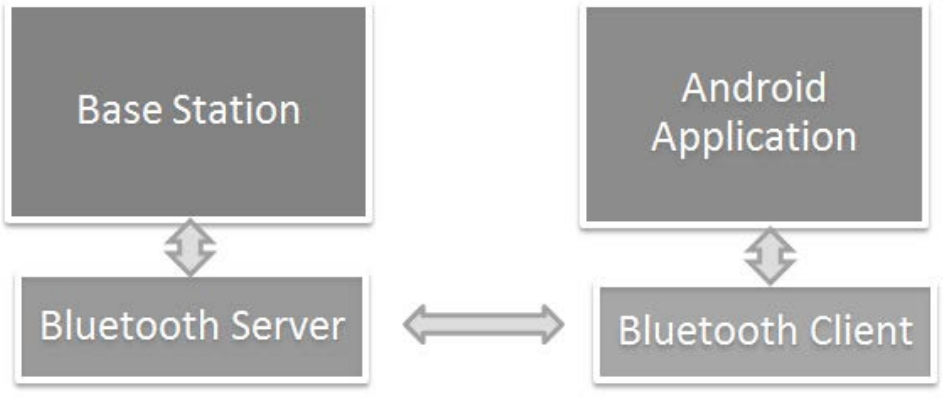


Fig. 6.4 Esquema comunicació bluetooth

Els clients són els que inicien la comunicació. Aquest busca tots els dispositius actius, emparellats o no, detecta la mac del servidor i li envia una petició per a connectar-se. Com ja s'ha dit anteriorment, ha d'haver-hi un únic servidor i

múltiples clients que s'hauran d'identificar mitjançant un codi. Aquest codi s'anomena UUID (veure[8]). L'UUID és un identificador universal de 128 bits.

Hi ha dues maneres diferents de realitzar aquesta comunicació:

1.- El mòbil busca tots els dispositius actius i els mostra una llista amb tots els que ha trobat per permetre a l'usuari escollir els dispositiu amb el qual es vol comunicar.
2.- El mòbil busca tots els dispositius actius i al detectar el dispositiu receptor desitjat inicia la comunicació sense interlocució de l'usuari.

L'opció escollida serà la segona ja que els dispositiu receptor sempre serà l'estació base, per tant, es definirà prèviament la mac del receptor. D'aquesta manera s'evitaran possibles problemes introduïts per l'usuari.

La comunicació és unidireccional ja que només és el dispositiu mòbil qui envia dades. Aquest dispositiu envia constantment el mateix missatge fins que l'usuari prem una altra tecla. D'aquesta manera no es necessari que l'estació base enviï un ACK com a confirmació de que el paquet ha arribat correctament.

Quan l'usuari surt de l'aplicació, automàticament s'envia un missatge de desconnexió i s'allibera per tal de permetre a qualsevol altre usuari un inici de la comunicació.

6.3. Leap Motion

6.3.1. Introducció

El controlador Leap Motion[1] és un petit dispositiu perifèric USB dissenyat per a ser col·locat sobre una superfície física, mirant cap amunt. L'ús de dues càmeres d'infrarojos monocromàtiques i tres LEDs infrarojos, permeten al dispositiu observar una àrea semiesfèrica, a una distancia d' aproximadament un metre.

Els LEDs generen un patró 3D de punts de llum IR i les càmeres generen prop de 300 *frame* per segon de dades obtingudes, que després són enviades a través del cable USB al ordinador, on són analitzades pel software del controlador Leap Motion, utilitzant "matemàtiques complexes", d'una manera que no ha sigut revelada per la empresa.

El software del Leap (veure [11]) analitza els objectes observats en el camp de visió del dispositiu. Reconeix mans, dits, i eines, informant tant de posicions, gestos i moviments.

[1] https://www.leapmotion.com/

El camp de visió del dispositiu és una piràmide invertida centrada en el dispositiu. El rang efectiu d'aquest s'estén des d' aproximadament 25 fins a 600 mil·límetres per sobre del dispositiu.

- Sistema de coordenades.

  El Leap utilitza un sistema de coordenades cartesianes destre. Els valor reportats estan en mil·límetres. L'origen està centrat al centre del controlador. Els eixos x i z es troben en el pla horitzontal, amb l'eix x paral·lel al costat llarg del dispositiu. L'eix y és vertical, amb el augment dels valors positius cap amunt. L'eix z té els valors positius que augmenten allunyant-se de la pantalla de l'ordinador.

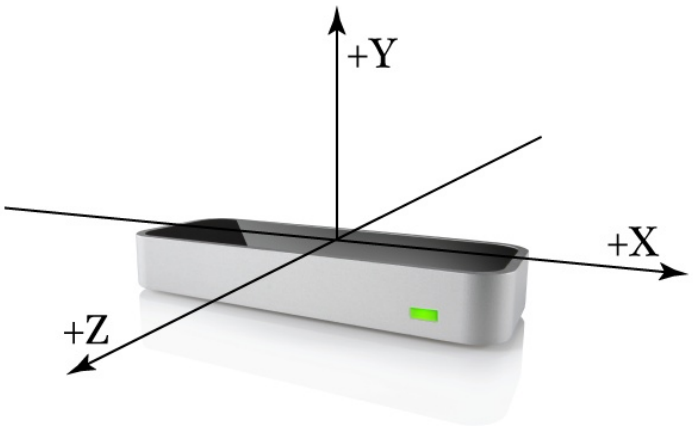


Fig. 6.5 Sistema de coordenades destre de Leap motion

Com el leap rastreja les mans, els dits i les eines en el seu camp de visió, proporciona actualitzacions com a conjunt, o *frame,* de les dades.

- *Frame*

  Cada *frame* conté llistes de les dades de seguiment bàsiques, tals com les mans, dits, i eines, així com gestos i els factors que descriuen el moviment global a l'escena.

  Quan es detecta una mà, dit, eina o un gest, el controlador li assigna un identificador únic. L'identificador segueix sent el mateix, sempre que aquesta entitat romangui visible dins del camp de visió del dispositiu. Si el seguiment s'ha perdut i recuperat, el Leap pot assignar un nou identificador (el software no sap que la mà o el dit és el mateix que el de l'anterior visió).

  El dispositiu analitza el moviment total que s'ha produït des de un *frame* anterior sintetitzant la translació, rotació i factor escalar, representats.

Tanmateix, utilitza tots els objectes dins del camp de visió en l'anàlisi del moviment. Si només es detecta una mà, basa els factors de moviment del *frame* en el moviment d'aquesta mà. Si es detecten dues mans, basa aquests factors en el de les dues mans conjuntes. També es poden obtenir els factors de moviment independents per a cada mà.

- Model de mà

  El model de la mà proporciona informació sobre la posició, les característiques, i el moviment de la mà detectada, així com les llistes dels dits i eines associades a la mà.

  De tota manera, no determina si la mà és esquerra o dreta. Poden aparèixer més de dues mans a la llista de mans del *frame*, si les mans de més d'una persona estan a la vista o es detecta un objecte com a mà. Es recomana tenir un màxim de dues mans en el camp de visió per a una òptima qualitat de seguiment del moviment.

  Els atributs *Direction* i *PalmNormal* són vectors unitaris de direcció que descriuen l'orientació de la mà respecte el sistema de coordenades del dispositiu. El vector *PalmNormal* apunta perpendicular a la mà i el vector *Direction* cap a endavant (**Fig. 6.6**).

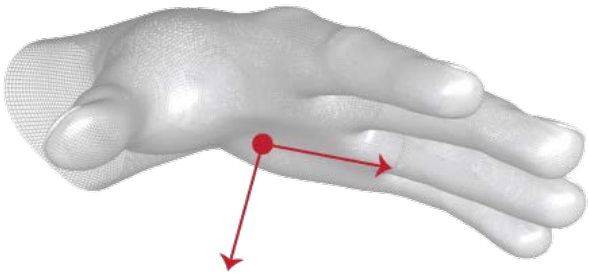


Fig. 6.6 Vectors *PalmNormal* i *Direction*

L'objecte *Hand* també proporciona diversos atributs reportant el moviment d'una mà detectada entre els *frames*. El Leap analitza el moviment de la mà, així com els dits i eines associats i reporta la translació, la rotació, i factors escalars, representats.

Aquests moviments es deriven comparant les característiques de la mà en el *frame* actual respecte les de un *frame* anterior.

- Model de dits i eines

  El dispositiu detecta i rastreja tant els dits com les eines dins del seu camp de visió; classifica els objectes segons la forma. Una eina és més llarga, més prima i més recte que un dit.

Les seves característiques físiques s'abstrauen en un objecte.

Les característiques *TipPosition* i *Direction* proporcionen les posicions de les puntes dels dits i les direccions en les que apunten aquests (**Fig. 6.7**).

Fig. 6.7 Vectors *TipPosition* i *Direction*

El que es vol aconseguir amb aquest dispositiu és el moviment del robot mitjançant el moviment de la mà. S'estudiaran diversos casos, i s'escollirà el cas amb el que s'obtingui un moviment més net i exacte.

La biblioteca de Leap Motion està escrita en C++. Leap Motion també utilitza una eina de codi obert per a generar enllaços de llenguatge per a C#, Java i Python.

Tots els arxius de la biblioteca, de codi i de capçalera necessaris per desenvolupar aplicacions i pluggins habilitats per al Leap s'inclouen en el SDK de Leap motion.

En aquest cas, el llenguatge escollit és C#, ja que proporcionarà una integració amb el projecte més senzilla.

6.3.2. Implementació

Per començar, s'utilitzarà l'exemple de codi proporcionat per la pàgina de Leap Motion. Un cop descarregat el codi corresponent per a C# (veure Annex III), és necessari afegir unes llibreries a la carpeta on es generi el projecte nou. Aquestes llibreries són Leap.dll, LeapCSharp.dll i LeapCSharp.NET4.0.dll.

Aquest exemple proporciona certa informació sobre les mans, com es pot veure en la **Fig 6.8**.

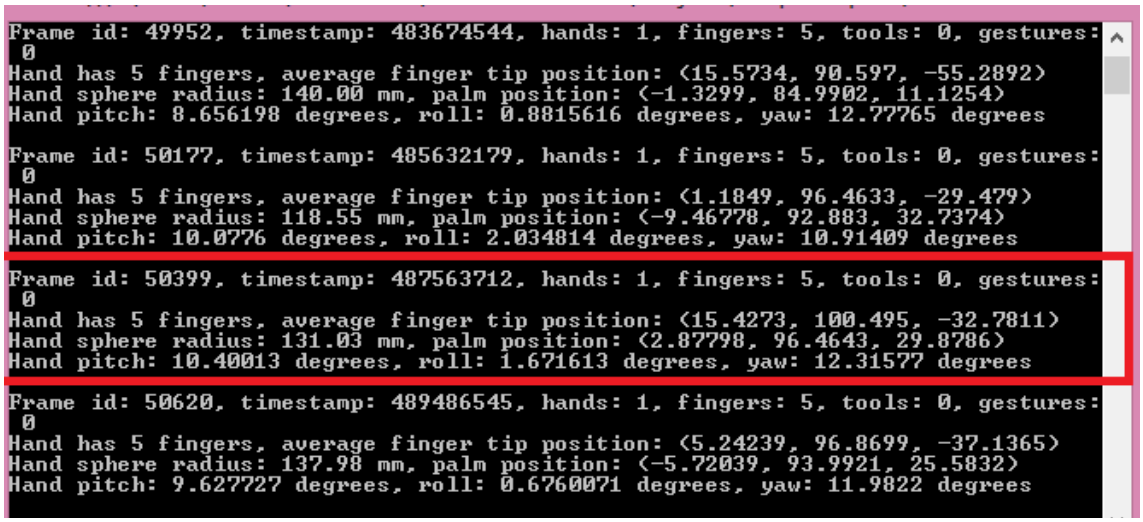


Fig. 6.8 Sample C#

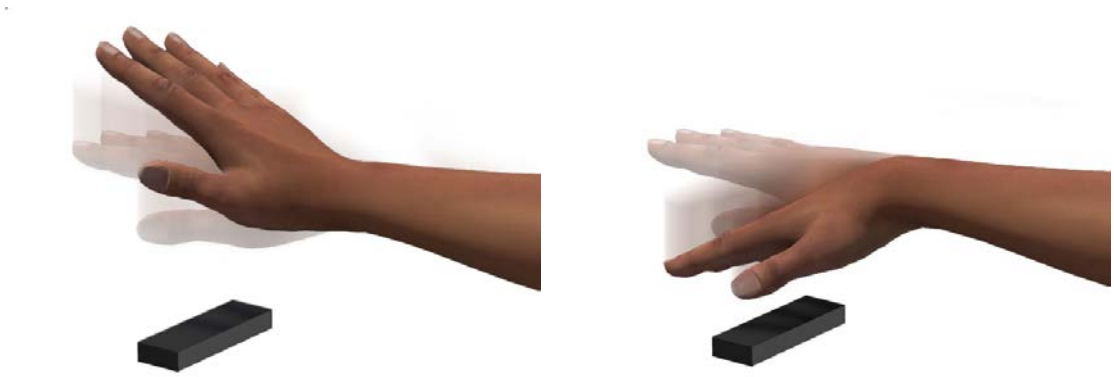
Aquest exemple retorna una paràgraf amb la informació següent:

- id del frame actual
- timestamp
- número de mans, dits, eines i gestos detectats
- vector de la posició mitja de la punta del dit
- radi de la esfera de la mà
- vector de la posició de la mà
- *Pitch*, *yaw* i *roll* de la mà (angles al voltant dels eixos x, y i z, respectivament)

A partir d'aquest exemple, es consideren els paràmetres que es poden utilitzar per al moviment del cotxe, i en conseqüència, el moviment de la mà més adequat.

La primera idea és utilitzar els angles *pitch*, *yaw*, i *roll*. Amb aquests paràmetres sí que influeix la mà utilitzada. Es decideix utilitzar la mà dreta, tot i que canviant els valors del paràmetres, es podria utilitzar l'altre mà.

D'aquesta manera el moviment de la mà seria el que s'observa a la **Fig 6.9**.



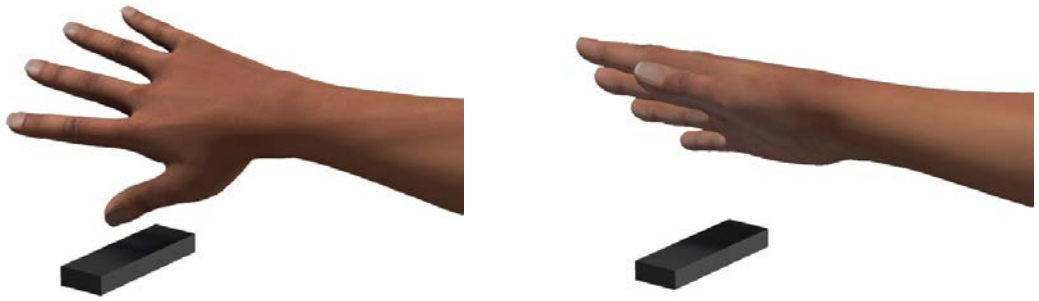


Fig. 6.9 Moviments mà amb Leap

.

A partir d'aquests moviments, s'estudien els valors d'aquests paràmetres més correctes. Finalment els valors escollits per a cada moviment del robot es mostren a **Taula 6.1**.

Taula 6.1 Límits per a cada moviment

Posició	Rang Pitch	Rang Roll
Quiet	-3<pitch<10	80<roll<122
Endavant	-60<pitch<-15	9<roll<75
Endarrera	20<pitch<50	125<roll<175
Dreta	-23<pitch<35	-180<roll<-100
Esquerre	-22<pitch<20	-100<roll<20

Un cop establerts els valors, es simulen els moviments del robot, obtenint un comentari de la direcció del moviment, per a cada moviment de mà.

Al provar moviments més continus, es troba el problema de la poca precisió que es té. Funciona correctament, però és fàcil confondre un moviment amb un altre, degut als paràmetres i valors utilitzats.

S'ha d'estar molt pendent del moviment de la mà i fer moviments molt exactes perquè no es confonguin uns amb altres.

Això no interessa ja que s'ha d'estar molt pendent de la mà i no dóna cabuda poder estar pendent del robot per controlar el camí a seguir. El que interessa és el contrari.

Per tant, la primera idea queda descartada i se'n buscaran de noves.

La idea següent és utilitzar la posició de la mà, que retorna un vector amb la distància del centre de la mà, respecte el centre del dispositiu. Amb aquest paràmetre, influeix la mà utilitzada. Es decideix utilitzar la mà dreta, tot i que canviant els valors del paràmetres, es podria utilitzar l'altre mà.

En aquest cas, el moviment de la mà seria el que s'observa a la **Fig 6.10**.

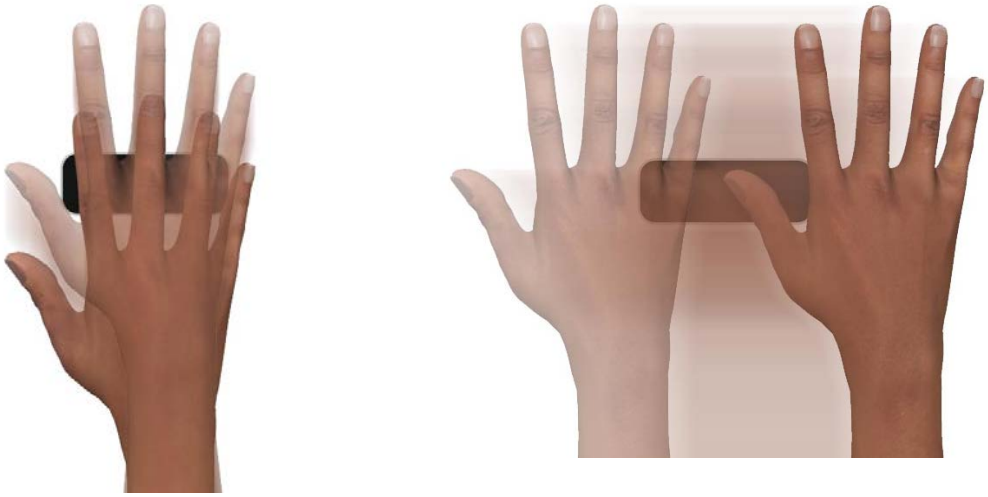


Fig. 6.10 Moviments mà amb Leap

A partir d'aquests moviments, s'estudien els valors d'aquests paràmetres més correctes. Finalment els valors escollits per a cada moviment es mostren en la **Taula 6.2**.

Taula 6.2 Límits per a cada moviment

Posició	Rang PalmPosition.x	Rang PalmPosition.z
Endavant	0<hand.PalmPosition.x<17	-72<hand.PalmPosition.z<-40
Endarrera	-23<hand.PalmPosition.x<-1	25<hand.PalmPosition.z<61
Dreta	19<hand.PalmPosition.x<80	1<hand.PalmPosition.z<21
Esquerre	-84<hand.PalmPosition.x<50	21<hand.PalmPosition.z<40

Un cop establerts els valors, es simulen els moviments del robot, obtenint un comentari de la direcció del moviment per a cada moviment de mà.
Un cop realitzada aquesta prova, s'observa que el resultat és semblant a l'anterior. És una opció poc exacte respecte l'objectiu desitjat. Per tant, aquesta idea també queda descartada.

Després de les dues proves fallides en les que el que s'utilitzava com a referència era la mà com a tal, es decideix investigar i provar per a una altre banda.

Es troba un paràmetre diferent, que és la translació. Aquesta retorna la magnitud i direcció del conjunt de l'objecte, en mil·límetres.
En aquest cas, l'objecte de referencia que s'utilitzarà és el *frame* no la mà com en els casos anteriors. Aquest proporciona un moviment global del camp de visió del Leap.

En aquest cas, el moviment de la mà serà el mateix que en la idea anterior. Es torna a mostrar a la **Figura 6.11**.

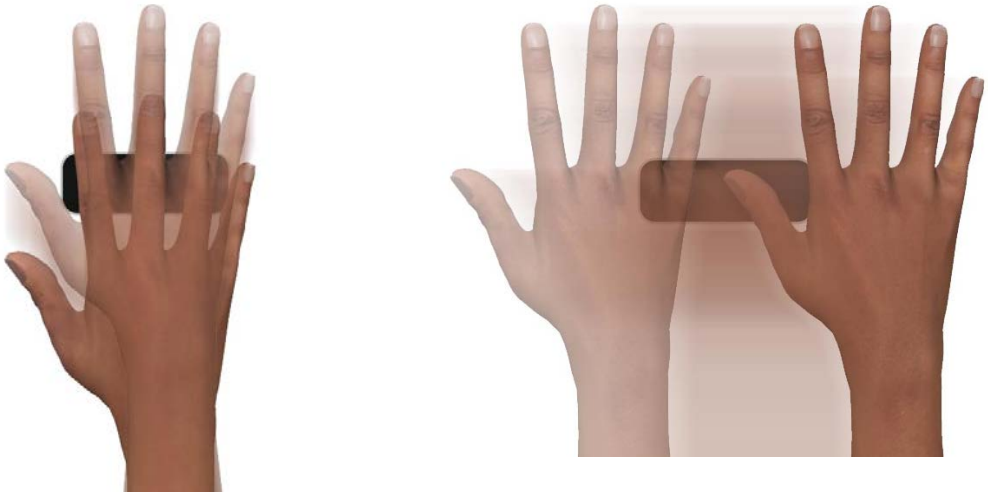


Fig. 6.11 Moviments mà amb Leap

S'estudien els valors del paràmetre més exactes per a cada moviment. Els valors escollits es mostren en la **Taula 6.3**

Taula 6.3 Límits per a cada moviment

Posició	Rang Translació
Endavant	translationZ < -20
Endarrere	translationZ > 25
Dreta	translationX > 25
Esquerre	translationX < -25

Un cop establerts els valors, es simulen els moviments del robot obtenint un comentari de la direcció del moviment per a cada moviment de mà.
Els resultats obtinguts en aquest cas s'aproximen molt al que es desitjava. Ja que és l'opció que millor ha funcionat, es decideix que serà la que s'aplicarà al moviment del robot, integrant-se en el projecte.

6.4. Conclusions

L'aplicació mòbil creada és una aplicació senzilla perquè el que es desitjava era aconseguir el control del robot. En un futur, es podria ampliar aquesta aplicació fent-la més complerta i afegint noves funcions.

En quant al dispositiu Leap Motion, s’ha aconseguit el objectiu desitjat, que era el moviment del cotxe a partir de diferents moviments de la mà.

CAPITOL 7. RESULTATS

Un cop explicats els nous mòduls implementats, s'iniciarà un procés de comparació per veure amb exactitud l'evolució del projecte.

A continuació, es realitzaran diverses proves per tal de fer visible aquesta evolució. Totes les proves es faran en el mateix circuit amb unes característiques ambientals similars. S'ha de recordar que un canvi en la superfície, en els camps magnètics o altres variacions podrien afectar al vehicle i, a la vegada, modificar el resultat de la prova.

Primer s'explicaran els canvis realitzats per a millorar el moviment del vehicle i, a continuació, les aportacions de control remot.

7.1 Resultats moviment del vehicle

Tal i com s'ha comentat al llarg del document, el principal objectiu és trobar mecanismes que permetin que el rover realitzi un moviment fluït.

Inicialment, el rover tenia un moviment poc atractiu. Realitzava els girs bruscament i no era capaç d'orientar-se ni saber si tornava a passar pel punt d'inici.

A la **Fig. 7.1** es mostra una prova prèvia a cap canvi.

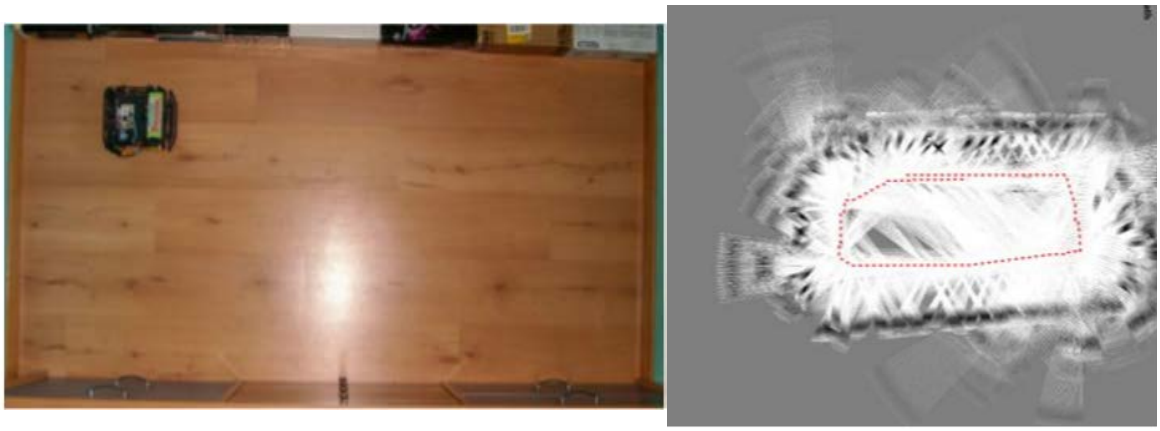


Fig. 7.1 Prova prèvia a cap canvi

La solució proposada per a millorar el moviment és la implementació del PID. Al Capítol 4, s'explica el procediment que s'ha seguit per a la realització dels dos casos en els que s'ha aplicat aquest controlador.

Per al seguiment de paret es necessiten uns valors, comentats al capítol, per identificar quan s'han de realitzar les correccions a partir de l'error. És a dir, un mínim i un màxim de l'error a partir del qual es realitzaran les correccions. S'han fet proves per determinar aquets valors i finalment s'han escollit els següents:

- S'ha d'establir un valor màxim fins al qual es realitzarà el seguiment de paret. Aquest valor serà 8 centímetres. Si l'error és més gran de 8 centímetres, es suposarà que hi ha una cantonada i es realitzarà un gir a l'esquerra. En canvi, si és més petit de 8 centímetres es tindran en compte dos casos en funció dels sensors laterals: si la distància del sensor superior és més gran que la del sensor inferior es considerarà que el robot s'està allunyant de la paret, i si la distància del sensor superior és més petita que la del sensor inferior, es considerarà que el robot s'està apropant. Tot això està explicat gràficament al capítol corresponent.

- L'error ha de ser més gran de 3 centímetres per a realitzar qualsevol correcció.

Per al gir a la dreta amb PID s'han d'establir un valor a partir del qual es començarà a realitzar el gir, que serà de 40 centímetres, i un altre a partir del qual es deixarà d'utilitzar el PID, que serà de 10 centímetres, ja que el robot estarà massa a prop de la paret.

A la **Fig.7.2** es mostra una prova amb el PID activat (dreta), i una altre sense (esquerra). Com es pot observar la imatge, amb el PID activat les cantonades són més arrodonides en comparació amb la imatge de l'esquerra.

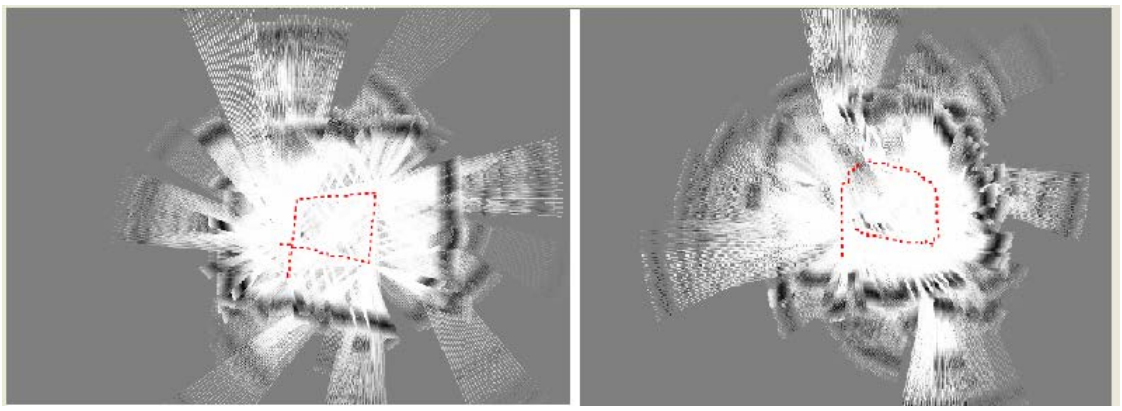


Fig. 7.2 Gir sense PID I amb PID

Una altre prova realitzada és la demostració del funcionament del PID, tant del seguiment de paret com del gir, demostrant alhora el bon funcionament del magnetòmetre calibrat amb un error mínim. (**Fig. 7.3**). A la imatge de l'esquerra es pot veure com el robot realitza alguna correcció a la part dreta degut a que s'havia apropat massa a la paret. Això és gràcies al seguiment de paret i al bon calibratge.

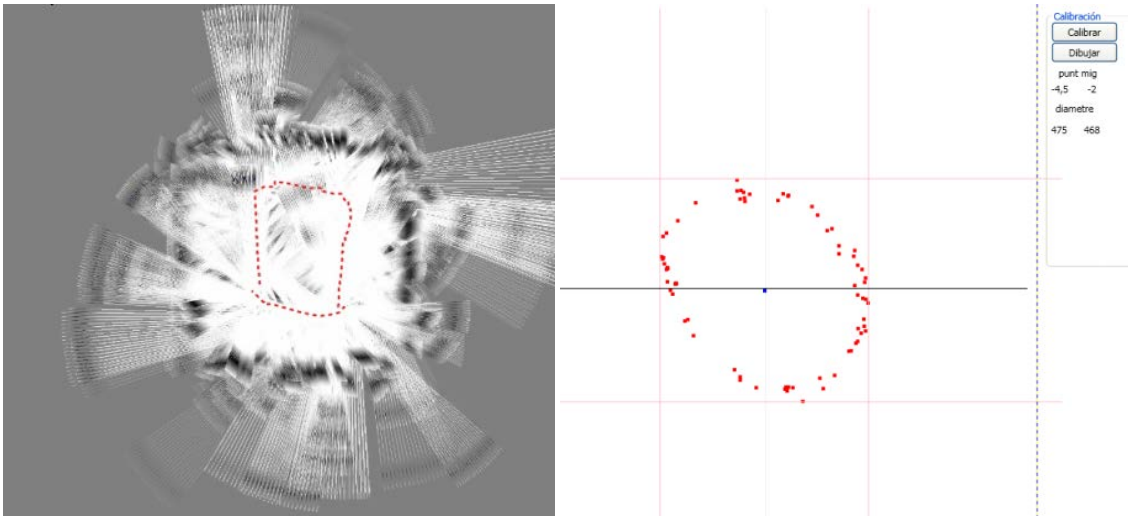


Fig. 7.3 PID i magnetòmetre

Tot i la prova realitzada anteriorment, se'n realitza una altre on, tot i que, el magnetòmetre està correctament calibrat, el robot realitza un gran nombre de correccions (**Fig. 7.4**).

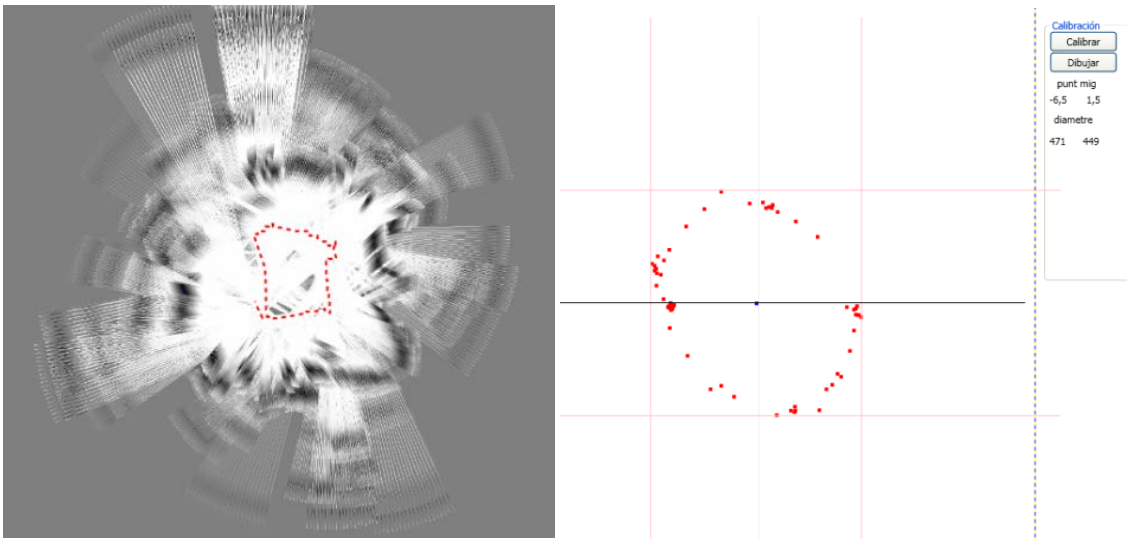


Fig. 7.4 Altes correccions amb magnetòmetre calibrat

7.2 Control remot

Les noves aportacions per al control remot del vehicle han estat la implementació del dispositiu Leap i la creació d'un aplicació mòbil Android.

Per aconseguir un moviment visiblement bo, tant el Leap com l'aplicació mòbil han hagut de passar per diverses fases i proves, ja que s'han de tenir en compte diversos factors com el temps de transmissió del paquet des del dispositiu fins a l'estació base, el temps de processat i el temps que tarda en arribar la informació al rover.

En el cas del Leap, s'han realitzat diverses proves amb diferents moviments de mà i diferents paràmetres del Leap utilitzats, explicats en el Capítol 6.
Finalment, s'ha trobat un resultat que ha sigut satisfactori ja que compleix amb els objectius esperats.

Per finalitzar, l'aplicació mòbil desenvolupada per dispositius android realitza les funcions bàsiques. La comunicació entre qualsevol dispositiu i l'estació base és via bluetooth. La base d'aquesta comunicació és ben simple: es connecten i es transmet constantment el mateix missatge fins que l'usuari prem un botó diferent. D'aquesta manera, si es perd algun missatge no es perd informació.

CAPITOL 8. CONCLUSIONS

8.1. Conclusions

Un cop finalitzat tot el treball, es considera que s'han assolit la major part dels objectius proposats inicialment. El moviment autònom del vehicle ha millorat considerablement amb la implementació del magnetòmetre i del pid.

Tal i com s'ha esmentat anteriorment, hi ha hagut problemes alhora d'afegir el Kinect ja que era necessària una placa molt potent per arrancar el sistema operatiu. El cost total del projecte hagués augmentat si s'hagués posat, tot i que, els resultats haguessin sigut molt positius.

Cal recalcar que les eines creades per a que l'usuari interactuï amb el robot són eines molt potents que si es fes un estudi a fons de cada una d'elles es podria aconseguir integrar molt més en el projecte.

Sobre l'objectiu de baix cost, la **Taula 8.1** mostra quin va ser el cost calculat de la primera versió del robot:

Taula 8.1 Cost primera versió

Components	Unitats	Preu (€/unitat)	Preu (€)
Estructura kit	1	71.43	71.43
Fez Panda	1	34.95	34.95
Motors DC	1	16.95	16.95
Sensors de distància	4	12.95	51.80
Mòdul xBee RF	2	16.99	33.98
Indicador de bateria	1	68.95	68.95
Total			**292.81**

En la segona versió, el cost ha augmentat a causa dels components afegits i dels components substituïts. A la **Taula 8.2** es mostra quins són aquests components i el preu final.

Taula 8.2 Cost final

Components	Unitats	Preu (€/unitat)	Preu (€)
Versió anterior			258.83
Mòdul xBee RF	2	19	38
Leap	1	58.52	58.52
Total			**355.35**

La primera versió no excedeix del pressupost fixat, 300€, que és extremadament baix si el comparem amb productes que hi ha al mercat (preus superior a 6000€ amb sensors d'alta qualitat o càmeres). Amb els canvis fets als robot, es supera el pressupost proposats però, tot i així, és econòmic.

Cal dir que, si en un futur, s'afegeix el Kinect aquest pressupost es veurà incrementat segons la versió del dispositiu que es compri.

Personalment, considerem que aquest projecte podria ser molt positiu pels alumnes si s'utilitzés a classe de programació ja que seria molt més dinàmic per ells i dóna una visió de la utilitat a la vida real.

8.2. Futures implementacions

Un cop finalitzat el treball sorgeixen noves idees les quals millorarien i el farien molt més atractiu. De les noves implementacions que es proposen algunes seran noves i d'altres seran extretes del projecte anterior i que no s'han aplicat al llarg d'aquest projecte.

Tot i que es pot observar una millora en el rover segueix sent un dels punts a millorar. Això es podria aconseguir amb uns sensors de major sensibilitat per a curtes distancies (menors de 40cm), amb una placa més potent i amb la introducció del kinect per a distàncies superiors a 40cm.

La implementació del kinect no provocaria el canvi dels motors ja que són prou potents per suportar el pes extra que s'afegiria però, sí que s'haurà de tenir en compte el consum de la bateria.

Un altre punt a millorar seria el pid. El pid només s'ha implementat quan el gir és cap a la dreta. El concepte per trobar a la fórmula correcta pel gir a l'esquerra és completament diferent perquè el sensor central no dona cap valor. Amb el kinect es podria solucionar aquest problema fent un processat de la imatge. Una de les llibreries més conegudes és OpenCV (*Open source Computer Vision library*). És una llibreria oberta desenvolupada per intel que proporciona un gran nombre de funcions per processar imatges. Per més informació, llegir annex II.

Una última proposta de millora és afinar l'aplicació mòbil. El front-end és bàsic encara que la comunicació està preparada per a que es pugui ampliar sense fer gaires canvis. En un futur és podria convertir el mòbil en una estació base on tot el processat de dades es faria al ordinador. Aquesta aplicació està pensada per mòbils android però es podria fer diverses adaptacions per a que es pogués aplicar a tablets i a mòbils amb altres sistemes operatius.

BIBLIOGRAFIA

[1] Alex Albala Diaz. “Indoor and Outdoor Rover with Simultaneous Localization and Mapping (SLAM)”. Tesis de Master. Univesitat Politecnica de Catalunya (UPC), 1 de desembre de 2011.

[2] Datasheet Magnetòmetre: http://www.alldatasheet.es/datasheet-pdf/pdf/406576/FREESCALE/MAG3110.html

[3] Informació PID: http://engineering.wichita.edu/esawan/gogoi.pdf

[4] Informació PID: http://en.wikipedia.org/wiki/PID_controller

[5] ZigBee: https://www.digi.com/technology/rf-articles/wireless-zigbee

[6] Mòduls XBee: http://www.xbee.cl/

[7] Definició Bluetooth: http://definicion.de/bluetooth/

[8] Informació sobre UUID:
http://developer.android.com/reference/java/util/UUID.html

[9] Desenvolupament comunicació en java:
http://developer.android.com/guide/topics/connectivity/bluetooth.html

[10] Iniciació desenvolupament aplicacions móvil en Android:
http://developer.android.com/training/basics/firstapp/index.html

[11] Leap Motion Develop C#:
https://developer.leapmotion.com/documentation/csharp/index.html

ANNEX I: Reinici del Visual

Una de les necessitats que s'ha observat durant el període de realització del projecte és que no es podia reiniciar la pantalla del visual. Per tornar a fer una prova s'havia de tornar a arrancar el programa. Per aquesta raó s'ha afegit una opció on es pogués reiniciar de manera fàcil.
Aquesta solució consta d'un botó que inicialitza tots els paràmetres necessaris: elimina totes les dades emmagatzemades del recorregut realitzat i el calibratge del magnetòmetre.

A **la Fig. I.1** es mostra una captura del programa al inici, i una altre del programa després de fer un reset.

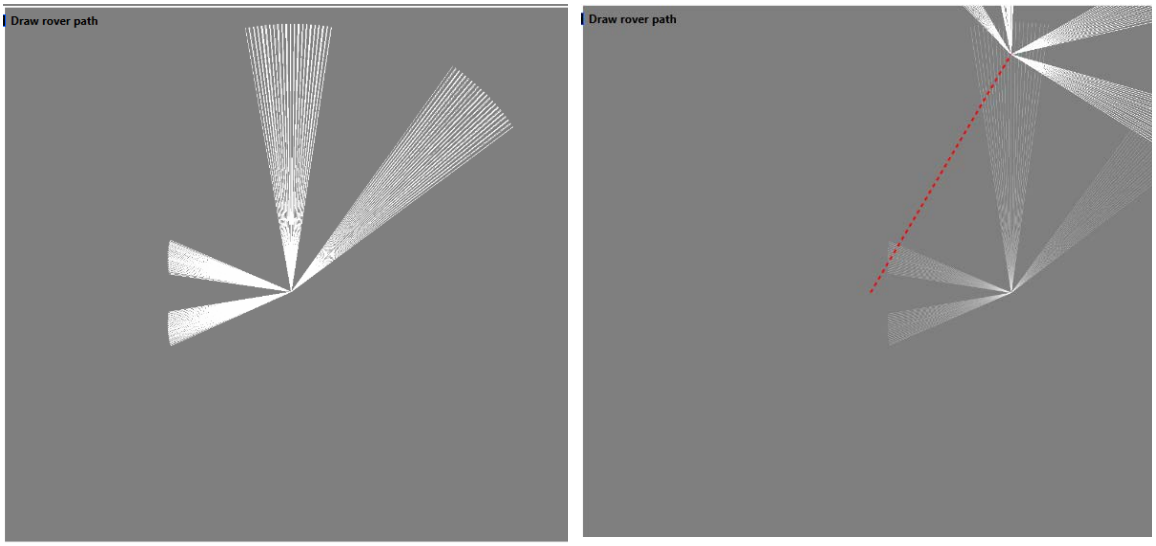


Fig. I.1 Exemple del reinici

Tal i com es pot observar, el punt inicial del recorregut ja no és el punt mig de la imatge. Això és degut a que el paràmetre és intern i només es pot actualitzat quan s'inicia el programa.

Ara bé, tot i no ser la millor opció, aquesta solució ha sigut de gran ajuda per a realitzar proves.

ANNEX II: LLIBRERIA OpenCV

En aquest annexa es farà una introducció d'una de les llibreries més conegudes per a la processar imatges.

OpenCV (Open source Computer Vision library) és una llibreria oberta per a processament d'imatges i visió computeritzada desenvolupada inicialment per Intel. La primer versió oberta va ser el 2006.

OpenCV permet:

a. Operacions bàsiques.
b. Processat d'imatges i anàlisis.
c. Anàlisis estructural.
d. Anàlisis de moviment.
e. Reconeixement del model.
f. Reconstrucció 3D i calibratge de la càmera.
g. Interfície gràfica.
h. Etc...

A la **Taula II.1**, s'expliquen els 4 mòduls que composen l'OpenCV.

Taula II.1. Mòduls de l'OpenCV

Mòdul	Descripció
cv	Conté les funcions principal de la biblioteca.
cvaux	Conté les funcions auxiliars
cxcore	Conté les estructures de dades i funcions de suport per àlgebra lineal
HighGUI	Funcions pel maneig del GUI

Alguns dels inconvenients d'aquest llibreria són el seguiment d'objectes ja que no ofereix un producte complert sinó que només algunes peces que t'ajuden a muntar el producte final. Como a inconvenient també es podria ressaltar la necessitat que té d'utilitzar la llibreria IPL per poder accedir a funcions de baix nivell.

ANNEX III: Sample C# Leap Motion

```
/******************************************************************
***************\
* Copyright (C) 2012-2013 Leap Motion, Inc. All rights reserved.
*
* Leap Motion proprietary and confidential. Not for
distribution.              *
* Use subject to the terms of the Leap Motion SDK Agreement
available at       *
* https://developer.leapmotion.com/sdk_agreement, or another
agreement         *
* between Leap Motion and you, your company or other
organization.              *
\******************************************************************
***************/
using System;
using System.Threading;
using Leap;

class SampleListener : Listener
{
     private Object thisLock = new Object ();

     private void SafeWriteLine (String line)
     {
          lock (thisLock) {
               Console.WriteLine (line);
          }
     }

     public override void OnInit (Controller controller)
     {
          SafeWriteLine ("Initialized");
     }

     public override void OnConnect (Controller controller)
     {
          SafeWriteLine ("Connected");
          controller.EnableGesture
(Gesture.GestureType.TYPECIRCLE);
          controller.EnableGesture
(Gesture.GestureType.TYPEKEYTAP);
          controller.EnableGesture
(Gesture.GestureType.TYPESCREENTAP);
          controller.EnableGesture
(Gesture.GestureType.TYPESWIPE);
     }

     public override void OnDisconnect (Controller controller)
     {
        //Note: not dispatched when running in a debugger.
          SafeWriteLine ("Disconnected");
     }
```

```
     public override void OnExit (Controller controller)
     {
          SafeWriteLine ("Exited");
     }

     public override void OnFrame (Controller controller)
     {
          // Get the most recent frame and report some basic
information
          Frame frame = controller.Frame ();

          SafeWriteLine ("Frame id: " + frame.Id
                    + ", timestamp: " + frame.Timestamp
                    + ", hands: " + frame.Hands.Count
                    + ", fingers: " + frame.Fingers.Count
                    + ", tools: " + frame.Tools.Count
                    + ", gestures: " + frame.Gestures ().Count);

          if (!frame.Hands.IsEmpty) {
               // Get the first hand
               Hand hand = frame.Hands [0];

               // Check if the hand has any fingers
               FingerList fingers = hand.Fingers;
               if (!fingers.IsEmpty) {
                    // Calculate the hand's average finger tip
position
                    Vector avgPos = Vector.Zero;
                    foreach (Finger finger in fingers) {
                         avgPos += finger.TipPosition;
                    }
                    avgPos /= fingers.Count;
                    SafeWriteLine ("Hand has " + fingers.Count
                          + " fingers, average finger tip
position: " + avgPos);
               }

               // Get the hand's sphere radius and palm
position
               SafeWriteLine ("Hand sphere radius: " +
hand.SphereRadius.ToString ("n2")
                         + " mm, palm position: " +
hand.PalmPosition);

               // Get the hand's normal vector and direction
               Vector normal = hand.PalmNormal;
               Vector direction = hand.Direction;

               // Calculate the hand's pitch, roll, and yaw
angles
               SafeWriteLine ("Hand pitch: " + direction.Pitch
* 180.0f / (float)Math.PI + " degrees, "
                         + "roll: " + normal.Roll * 180.0f /
(float)Math.PI + " degrees, "
```

```
                        + "yaw: " + direction.Yaw * 180.0f /
(float)Math.PI + " degrees");
          }

          // Get gestures
          GestureList gestures = frame.Gestures ();
          for (int i = 0; i < gestures.Count; i++) {
               Gesture gesture = gestures [i];

               switch (gesture.Type) {
               case Gesture.GestureType.TYPECIRCLE:
                    CircleGesture circle = new CircleGesture
(gesture);

                   // Calculate clock direction using the angle
between circle normal and pointable
                    String clockwiseness;
                    if (circle.Pointable.Direction.AngleTo
(circle.Normal) <= Math.PI / 4) {
                         //Clockwise if angle is less than 90
degrees
                         clockwiseness = "clockwise";
                    } else {
                         clockwiseness = "counterclockwise";
                    }

                    float sweptAngle = 0;

                   // Calculate angle swept since last frame
                    if (circle.State !=
Gesture.GestureState.STATESTART) {
                         CircleGesture previousUpdate = new
CircleGesture (controller.Frame (1).Gesture (circle.Id));
                         sweptAngle = (circle.Progress -
previousUpdate.Progress) * 360;
                    }

                    SafeWriteLine ("Circle id: " + circle.Id
                              + ", " + circle.State
                              + ", progress: " +
circle.Progress
                              + ", radius: " + circle.Radius
                              + ", angle: " + sweptAngle
                              + ", " + clockwiseness);
                    break;
               case Gesture.GestureType.TYPESWIPE:
                    SwipeGesture swipe = new SwipeGesture
(gesture);
                    SafeWriteLine ("Swipe id: " + swipe.Id
                              + ", " + swipe.State
                              + ", position: " + swipe.Position
                              + ", direction: " +
swipe.Direction
                              + ", speed: " + swipe.Speed);
                    break;
               case Gesture.GestureType.TYPEKEYTAP:
```

```
                        KeyTapGesture keytap = new KeyTapGesture
(gesture);
                        SafeWriteLine ("Tap id: " + keytap.Id
                                 + ", " + keytap.State
                                 + ", position: " +
keytap.Position
                                 + ", direction: " +
keytap.Direction);
                        break;
                  case Gesture.GestureType.TYPESCREENTAP:
                        ScreenTapGesture screentap = new
ScreenTapGesture (gesture);
                        SafeWriteLine ("Tap id: " + screentap.Id
                                 + ", " + screentap.State
                                 + ", position: " +
screentap.Position
                                 + ", direction: " +
screentap.Direction);
                        break;
                  default:
                        SafeWriteLine ("Unknown gesture type.");
                        break;
                  }
            }

            if (!frame.Hands.IsEmpty || !frame.Gestures
().IsEmpty) {
                  SafeWriteLine ("");
            }
      }
}

class Sample
{
      public static void Main ()
      {
            // Create a sample listener and controller
            SampleListener listener = new SampleListener ();
            Controller controller = new Controller ();

            // Have the sample listener receive events from the
controller
            controller.AddListener (listener);

            // Keep this process running until Enter is pressed
            Console.WriteLine ("Press Enter to quit...");
            Console.ReadLine ();

            // Remove the sample listener when done
            controller.RemoveListener (listener);
            controller.Dispose ();
      }
}
```